المختصر المفيد في لغة PHP

اعداد : أحمد إبراهيم

الفهرس

المقدمة 3

البداية مع php 5

الفصل الأول (المتغيرات variables) 6

الفصل الثاني (عبارات الشرط والتكرار) 14

الفصل الثالث (العلامات المنطقية) 19

الفصل الرابع (المصفوفات) 21

الفصل الخامس (الوقت والتاريخ) 34

الفصل السادس (الدوال الرياضية والتعامل مع الأرقام) 37

الفصل السابع (تعريف واستدعاء الدوال) 38

الفصل الثامن (استخدام JSON لتخزين وجلب البيانا) 41

الفصل التاسع (رفع الملفات إلى الخادم) 44

الفصل العاشر (دوال منوعة) 49

الفصل الحادي عشر (دوال الأمن والحماية) 78

الفصل الثاني عشر (دوال التعامل مع الصور) 62

الفصل الثالث عشر (دوال التعامل مع سيرفر FTP) 71

الفصل الرابع عشر (اشتمال الملفات والتعامل معها) 75

الفصل الخامس عشر (تصيد وتتبع الأخطاء) 91

الفصل السادس عشر (session & cookies) 103

الفصل السابع عشر (MYSQL) 106

الفصل الثامن عشر (تعليمات sql) 110

الفصل التاسع عشر (دوال التعامل مع MYSQL) 118

الفصل العشرون (البرمجة كائنية التوجه OOP) 126

ملاحظات عامة 139

المقدمة

كيف فكرت بانشاء هذا الكتاب ؟

فكرت بتعلم تصميم مواقع الإنترنت وأنا لم يكن لي أي خلفية مسبقة في البرمجة ولست صاحب أختصاص في هذا المجال وانا مجرد هاوي أعشق البرمجة ولازلت أتعلم الى حد هذه اللحظة عندما بدأت أتعلم البرمجة (ولا ازال) أحببتها بشكل جنوني وأصبحت البرمجة تأخذ كل وقتي حيث أصبحت أمضي في اليوم من 10 الى 15 ساعة امام شاشة اللابتوب وفي بعض الأحيان استمر لأكثر من 30 ساعة بشكل متواصل , في البداية أكبر مشكلة واجهتني هي أنه لا توجد كتب ضخمة وتشمل كل جوانب لغة البرمجة والمحتوى العربي ضعيف للغاية والمشكلة الأكبر هي أني لا أجيد اللغة الإنجليزية بشكل جيد لذا تقريباً قرأت أكثر الكتب العربية التي كتبت في هذه اللغة وبما أن أكثرها هو عبارة عن مجموعة صفحات ليس إلا وأغلب المعلومات التي فيه هي فقط أساسيات اللغة وعند قراءة كتاب تجد إن كل مافيه تقريباً هو موجود في كتاب آخر مع إضافة بسيطة لذا فكرت بأن أجمع كل معلومة جديدة أتعلمها في هذه اللغة وادونها بشكل مختصر ليسهل علي الوصول لها عند نسيانها لذا أنشأة مرجع بسيط خاص بي لهذ الغة , وبعد فترة أصبح فيه مجموعة لا بأس بها من المعلومات ففكرت أن أنشرها بشكل كتاب ليستفاد منها كل من يحب تعلم لغة الـ php , لذا قد تجدون في هذا الكتاب الكثير من الإختصار في شرح الأوامر ولم أضع أمثلة مطوله بل فقط أمثلة بسيطة لتصل الفكرة .

وبنفس الطريقة أنشأت مراجع خاصة بي تشمل عدة لغات وتقنيات منها مرجع للغة HTML ومرجع للغة CSS ومرجع للغة JavaScript وسأنشر هذه الكتب قريباً لتكون متاحة للجميع .

ملاحظة/ هذه هي الطبعة الأولى من الكتاب طبعت بتاريخ 2014/4/20 ومن المحتمل أن أضيف على هذا المرجع الصغير بعض الإضافات وأجري عليه بعض التعديلات وأنشره كطبعه ثانية (في المستقبل) .

ارحب بكل أسئلتكم واستفساراتكم أيضاً كل أقتراحاتكم لأنه كما قلت سابقاً إني لازلت في بداية طريقي في البرمجة ولا زلت اتعلم شيء جديد كل يوم .

يمكنكم التواصل معي على العنوان التالي :- ahmedcia17@gmail.com

نظرة عامة عن اللغة

php هي لغة برمجة لإنشاء تطبيقات وصفحات مواقع الإنترنت وأن php هي إختصار لـ (Personal Home Pages) وتعني الصفحات الشخصية , Rasmus Lerdof هو من أنشأ لغة php وفي البداية كانت لغة php لغة بسيطة لكن مما ساهم في توسعها هو ان صاحبها قد طرحها على الإنترنت بشكل مجاني وجعلها مفتوحة المصدر أي أنه يمكن لأي شخص التعديل أو الإضافة عليها , ومن فترة إنشائها الى الآن تم إضافة الكثير من الإضافات عليها من قبل العديد من المبرمجين فاصبحت لغة php لغة قوية وفيها الكثير من الميزات حيث أنها تعتبر من اللغات سهلة التعامل وسريعة في تنفيذ البرامجيات ولغة آمنه وهي تعمل على جميع أنضمة التشغيل بخلاف الكثير من لغات البرمجة الأخرى ,وكل ذلك ساهم بأنتشار هذه اللغة إنتشاراً واسعاً حيث يعتقد البعض إن 75% من مواقع الإنترنت مصممة بهذه اللغة ومن بين هذه المواقع موقع التواصل الإجتماعي Facebook , لغة php تعتبر لغة من طرف الخام Server حيث أن معالجها يوجد على الخادم أي أنك إذا كتبت برنامج بهذه اللغة وحاولت تشغيله مباشرتاً من المتصفح فإنه لن يعمل لأنك يجب أن ترفع هذا البرنامج على خادم على الإنترنت وبما أن هذه العملية ستكون بطيئة إذ أننا نحتاج الى تجربة البرنامج الذي نكتبه والتعديل عليه مراراً وتكراراً قبل أن يكتمل لذا توجد هناك العديد من البرامج التي تسمى سيرفر محلي وإن هذا البرنامج هو عبارة عن سيرفر (خادم) يتم تنصيبه على جهازك الشخصي وبهذا يمكنك تشغيل ما تكتبه بلغة الـ php بدون الحاجة للأتصال بالإنترنت ومن هذه البرامج AppServ و Apache Server و Xamp وغيرها العديد ويمكنك أن تنزل واحداً من الإنترنت وتنصبه على جهازك لتتمكن من العمل بسهوله مع php , أنا استخدم AppServ وهوبرنامج سهل وجميل جداً .

ملاحظة/ يفترض بالقارئ أن يعرف كيفية التعامل مع لغة html قبل أن يقراء هذا الكتاب .

أحمد إبراهيم

البداية مع php

إن لغة php تبدأ بالوسم (<?php) ووسم النهاية هو (?>) وفي داخل هذين الوسمين يكتب كود البرنامج , ويمكن كتابة وسم البداية بهذا الشكل (<?)أي من دون كلمة php لكن في هذه الحالة يمكن أن يتداخل هذا النص مع xml لذا فمن الأفضل كتابة وسم البداية بهذه الطريقة (<?php) .

وهناك عدة دوال تستخدم مع لغة php ومن هذه الدوال :-

- **الدالة echo**

وظيفة هذه الدالة (echo) طباعة (عرض) ما يكتب داخلها

مثال :-

```
<?php
echo "welcome";
?>
```

كما لاحظت فأن النص المدخل يكتب بين علامتي تنصيص (" ") , حيث إننا نكتب هذين العلامتين في أي مكان نريد أن نضع نصاً وذلك لكي نخبر المحلل إن هذه قيمة نصية , ولكن إن أردنا إن نكتب قيمة عددية فلا نحتاج أن نكتب علامتي التنصيص , وفي نهاية السطر نضع فارزة منقوطة (;) حيث إن هذه الفارزة المنقوطة توضع في نهاية كل سطر في هذه اللغة وذلك لكي نخبر المحلل عن انتهاء سطر وبدأ سطر جديد , وإن نسيت وضع هذه الفارزة المنقوطة في نهاية أي سطر فسوف يحدث خطأ .

ملاحظة/ الفواصل بين الدالة والقيمة في لغة php (في كل دوال اللغة) لا تؤثر على البرنامج , لاحظ هذين المثالين حيث لا يوجد فرق بينهما :-

```
echo "welcome";
echo          " welcome "     ;
```

الفصل الأول
(المتغيرات Variables)

إن المتغير يمكن أن يعطى أي اسم وأيضا يمكن إعطائه أي قيمة ولكي يتعرف المحلل على إن هذا متغير يجب وضع علامة دولار ($) قبل اسم المتغير , ولاحظ أن اسم المتغير حساس لحالة الأحرف الكبيرة والصغيرة حيث إن M ليست هي نفسها m , يمكن أن يتكون اسم المتغير من الحروف الانجليزية الكبيرة والصغيرة ومن أرقام (9 - 0) ولكن لا يمكن أن يكون اسم المتغير فقط أرقام ويمكن أن تدخل رمز واحد فقط في اسم المتغير وهذا الرمز هو (_) مثل $man_5 , لاحظ أن الرمز (_) يختلف عن الرمز (-) .

مثال :-

```
$Ahmed = 50 ;

$Ali = " welcome ";

echo $Ahmed ;

echo $Ali ;
```

كذلك يمكن كتابة نص ومتغير في داخل الدال **echo** , مثال :-

```
$value = "welcome" ;

echo " my text is $value" ;
```

لتكون النتيجة (my text is welcome)

ملاحظة/ يمكن إنشاء متغير اسمه قيمة متغير غيرة وذلك بإضافة العلامة ($) قبل اسم المتغير , ويستفاد من هذه الطريقة إذا أريد ربط متغير معين بمتغير آخر مثل الاسم وكلمة السر .

مثال :-

```
$a = "sager" ;

$$a = "My program" ;
```

لاحظ هنا إننا أنشأنا متغير جديد اسمه sager وقيمته My program .ولطباعة هذا المتغير الجديد يمكن أن نكتب أي صيغة من هذه الصيغ الثلاثة :-

```
echo $$a ;
echo ${" sager "} ;
echo ${"$a"} ;
```

* يمكن إنشاء متغير فارغ وذلك بعدم إسناد أي قيمة له

```
$X ;
```

او يمكن أن نعطيه قيمة فارغة وذلك بإسناد القيمة null والتي هي تعبر عن اللاشيء (القيمة الفارغة)

```
$x=null ;
```

المتغيرات الثابتة

وهي عبارة عن متغيرات يتم التصريح عنها كمتغيرات عامة على مستوى الصفحة كاملة , والفرق بينه وبين المتغير العادي إنه لا يسبق بعلامة دولار ($) عند كتابة الاسم وينطبق على الاسم ما ينطبق على المتغير العادي من حروف وأرقام وعلامة (_) ويستخدم هذا النوع من المتغيرات في حفظ المتغيرات الهامة على مستوى صفحة أو برنامج كامل ويتم فتح هذا المتغير عن طريق الدالة (**define**) والقاعدة العامة لها :-

```
define ( قيمة المتغير , اسم المتغير );
```

مثال :-

```
define ( man ,"Ahmed");
echo man ;
```

أنواع المتغيرات

أولاً :- المتغيرات الرقمية أو العددية (Integers)

يتم تعريف المتغيرات الرقمية بمجرد إعطاء المتغير قيمة رقمية وبدون علامات اقتباس (" ") .

مثال :-

```
$x = 5 ;
```

ويمكن إجراء عمليات الجبر الرياضية عليه , مثال :-

```
$x = 5 ;
$y =6 ;
$z = $x + $y ;
echo $z ;
```

جمع	+	Echo 2+2;
طرح	-	Echo 4-2;
قسمة	/	Echo 4/2;
باقي القسمة	%	Echo 7%2
ضرب	*	Echo 2*2;
إضافة رقم واحد على المتغير	++	$x++
طرح رقم واحد من المتغير	--	$x--

لاحظ أسبقية العمليات الرياضية :-

أولاً :- الأقواس .

ثانياً :- الضرب ثم القسمة .

ثالثاً :- الطرح ثم الجمع .

العمليات الرياضية على المتغيرات

1- متغير أسي حيث أن E يمثل رمز الأس $x= 3E10 ;

2- إجراء عملية جبر رياضية على قيمة المتغير القديمة وإضافة العملية الجديدة عليه

مثال/

```
$x=5;
$x * = 4 ;
echo $x ;
```

هنا ستكون النتيجة 20

ثانياً :- المتغيرات النصية (string)

هذا النوع من المتغيرات يقبل تخزين نصوص وأرقام ولكن بوضعها بين علامتي تنصيص (" ").

مثال :-

```
$y = " welcome";
```

ثالثا :- المتغيرات المنطقية (Boolean)

وهذا المتغير تكون له قيمتان أما true أو false ونستطيع كتابة هذا المتغير فقط بإضافة إحدى القيمتين إلى المتغير .

مثال :-

```
$y = true ;
$x = false ;
```

دوال المتغيرات

- **الدالة intval**

تعمل هذه الدالة على تحويل المتغير بجميع أحواله إلى متغير صحيح , فإذا كاننصاً ويوجد به رقم يتم استخراج الرقم منه .

مثال :-

```
echo intval ("504Man");          //504
echo intval (50.6);              //50
```

- **الدالة floatval**

تعمل هذه الدالة للحصول على أو استخراج الأرقام العشرية من المتغيرات .

مثال :-

```
echo floatval ("60.7Man");          //60.7
```

- **الدالة doubleval**

هذه الدالة تحول المتغير إلى نوع الرقمي المضاف من أي نوع آخر .

مثال/

```
$x="arab 50.9 program" ;
$y=doubleval( $x ) ;
print $y ;
```

تكون النتيجة 50.9

- **الدالة gettype**

تعمل هذه الدالة على إخراج قيمة هي عبارة عن نوع المتغير أما نصي string أو رقمي (عددي) integer أو منطقي Boolean , وذلك بواسطة تمرير المتغير على الدالة .

مثال :-

```
$y =10 ;
echo gettype ($y);
```

حيث سيتم الحصول على نوع المتغير وهو integer وطباعته

- **الدالة var_dump**

تقوم هذه الدالة بتحديد نوع المتغير والبيانات الموجودة فيه .

مثال/

```
var_dump( $x ) ;
```

- **الدالة settype**

هذه الدالة تعمل على تحويل نوع المتغير الممرر إليها إلى أي نوع آخر .

مثال :-

```
$Bol = false ;
echo settype ( $Bol , "string" ) ;
```

حيث يمكن وضع integer ليكون رقم

- **الدالة unset**

هذه الدالة تعمل على إزالة وجود المتغير من الصفحة , تستخدم هذه الدالة في الحماية أحيانا وأحياناً أخرى في تسريع عمل الصفحات .

مثال :-

```
$y = "value" ;
echo $y ;
unset ( $y ) ;
echo $y ;
```

النتيجة إنه ستظهر القيمة value مرة واحدة فقط لأنه أزيل بعد العرض الأول .

- **الدالة isset**

تستعمل هذه الدالة لمعرفة إذا كان هذا المتغير تم إنشائهمسبقاً أم لم يتم إنشاؤه , وهي لا تتطلب سوى اسم المتغير الذي تريد فحص وجوده وسوف ترجع القيمة (1) إذا تم إنشاؤه ولا ترجع شيء إذا لم يكن المتغير موجود .

مثال :-

```
$n = "Man" ;
echo isset ( $n ) ;
```

- **الدالة is_bool**

تبين هذه الدالة هل المتغير من نوع Boolean إذا كان كذلك تعيد القيمة true .

مثال/

```
$x=false ;
if( is_bool( $x ) ) { print "Boolean" ; }
```

- الدالة **is_double**

تبين هذه الدالة هل المتغير من النوع double إذا كان كذلك تعيد القيمة true .

- الدالة **is_int**

تبين هذه الدالة هل المتغير من النوع integer إذا كان كذلك تعيد القيمة true .

- الدالة **is_numeric**

تبين هذه الدالة هل المتغير من نوع رقمي إذا كان كذلك تعيد القيمة true .

الفصل الثاني
(عبارات الشرط والتكرار)

أولاً :- العبارات الشرطية

1- العبارة الشرطية (if)

الصيغة العامة لها هي :-

```
if ( حالة الشرط ) { الأمر الذي سينفذ عند تحقق الشرط }
```

هذه العبارة تعني إذا تحقق الشرط نفذ هذا الأمر .

مثال/

```
$x =10 ;
if ( $x == 10 ) { echo " تساوي " ; }
```

* لاحظ الفارزة المنقوطة لا توضع في حالة الشرط .

إليك هذا الجدول يوضح علامات المقارنة :-

الأداة	الوصف
= =	لمساواة القيم فقط
= = =	لمساواة القيم ونوع المتغير
<	أصغر من
>	أكبر من
< =	أصغر من أو يساوي
> =	أكبر من أو يساوي
< >	لا يساوي
!=	عدم المساواة في القيمة فقط
!= =	عدم المساواة في القيمة والنوع

2- العبارة الشرطية (if else)

وهي تشبه عبارة if لكن هنا في حالة عدم تحقق الشرط سينفذ أمر آخر بعد else .

الصيغة العامة لها هي :-

```
if ( حالة الشرط ) { أمر تحقق الشرط } else { أمر عدم تحقق الشرط }
```

مثال /

```
$x = 5 ;
if ($x > 20 ) { eho "أكبر " ; }
else { echo " أصغر " ; }
```

ملاحظة/ يمكن استخدام العبارة الشرطية (if) في داخل عبارة شرطية ثانية لاحظ هذا المثال :-

```
$x = 10 ;
if ( $x > 10 ) { echo " Big " ; }
else if ( $x < 10 ) { echo " small " ; }
else { echo " welcome " ; }
```

ويمكن أن نضيف عبارات كما نشاء , كذلك يمكن تعشيش العبارات الشرطية .

مثال /

```
$x = " Ahmed " ;
$y = 50 ;
if ( $x == " Ahmed " ) { if ( $y = 50 ) { echo " الاسم والرقم صحيحان " ; }
else { echo " فقط الرقم غير صحيح " ; } }
else { echo " اسم تسجيل الدخول خاطئ " ; }
```

3- عبارة الشرط if المختصرة (?)

عبارة الشرط هذه هيتماماً نفس العبارة السابقة if لكنها في العادة تستخدم للشروط القصيرة المكونة من سطر واحد ويمكن أن نضع بعدها نقطتين لتكون بمثابة else , وتكون الصيغة العامة لها بهذا الشكل

```
الشرط ? كود تحقق الشرط : كود عدم تحقق الشرط ;
```

4- العبارة الشرطية (switch)

وتستخدم هذه العبارة الشرطية للتحقق من عدة شروط وتنفيذ أمر خاص بكل شرط وإن أردنا استخدام العبارة **if** فإننا نحتاج إلى تكرار هذه العبارة عدة مرات ولتلافي هذا التكرار نستخدم العبارة (**switch**).

الصيغة العامة لها هي :-

```
switch ( القيمة ) {
case قيمة ;        <--- التحقق من القيمة
تنفيذ أمر ;        <--- ينفذ إذا كان الشرط متحقق
break ;            <--- إغلاق case
case قيمة ;
تنفيذ أمر ;        <--- وهنا تعاد العملية ويمكن الإعادة أكثر من مرة
break ;

default:           <--- إذا لم تتحقق أي قيمة من القيم السابقة
تنفيذ أمر ;
break;
}
```

مثال :-

```
$x =2 ;
switch ( $x ) { case 1;
 $y = " واحد " ;
break ;
 case 2;
 $y = " اثنان " ;
break ;
 case 3;
 $y = " ثلاثة " ;
break ; }
```

ثانياً :- عبارات التكرار

1- عبارة التكرار (for)

وهي تستعمل لتكرار عملية معينة عدة مرات .

الصيغة العامة لها هي :-

```
for ( قيمة المتغير ; الشرط ; حالة تحقق الشرط ) { تنفيذ أمر أو مجموعة أوامر }
```

مثال :-

```
for ($x = 1; $x < 5 ; $x++) { echo $x . "<br>"; }
```

ستكون النتيجة هكذا :-

```
1
2
3
4
```

2- عبارة التكرار (while)

الصيغة العامة لها هي :-

```
while ( الشرط ) { تنفيذ أمر أو مجموعة أوامر }
```

مثال :-

```
$x = 1 ;
while ( $x < 5) { $x++ ; echo $x . "<br>"; }
```

ستكون النتيجة هكذا :-

```
2
3
4
5
```

* لاحظ الفرق بين العبارة **for** والعبارة **while** .

3- عبارة التكرار (do while)

حيث في هذه الحلقة سوف يتمأولاً تنفيذ الكود وبعدها يتم التحقق من الشرط , والصيغة العامة لها هي :-

```
do { كود } while ( الشرط ) ;
```

الفصل الثالث

العلامات المنطقية (! , xor , or , and)

إن هذه العلامات المنطقية تتيح لك تنفيذ الكود بعد التحقق من مجموعة شروطوأيضاً تنفيذ الكود إذا تحققت كل الشروط (**and**) أو تحقق شرط واحد أو كلها (**or**) ويمكن التحقق من صحة شرط واحد فقط ولا يصح تحقق أكثر من شرط (**xor**) ويمكنأيضاً التحقق من عدم صحة شيء لكي تعمل على تنفيذ شيء آخر (**!**)

1- المعامل (and) ونظيره (&&)

يمكن استعمال (**and**) أو (**&&**) للتحقق من صحة عدة شروط في وقت واحد .

مثال :-

```
$x = 10 ;
$y = 5 ;
if ( $x == 10  &&  $y == 6 ) { echo " Hi " ;}
```

نلاحظ بأنه لن يظهر شيء .

2- المعامل (or) ونظيره (||)

يمكن استعمال (**or**) أو (**||**) للتحقق من صحة شرط واحد أو كل الشروط الموجودة .

مثال :-

```
$x = 10 ;
$y = 5 ;
if ( $x == 10  or  $ == 6 ) { echo " Hi " ; }
```

تلاحظ بأنه سينفذ الشرط .

3- المعامل (xor) ونظيره (^)

يمكن استعمال xor او ^ للتحقق من صحة شرط واحد من بين عدة شروط ولا يصح أن يتحقق أكثر من شرط

```
$x = 10 ;

$y = 5 ;

if ( $x == 10  xor  $y == 6 ) { echo " Hi " ; }
```

4- المعامل (!)

لا يمكن استخدام (Not)أبداً لأنها غير موجودة في لغة php لكن يمكن استخدام نظيرها (!) وله نفس وظيفة (Not) حيث يتأكد من أن هناك قيمة غير صحيحة (false) لكي يتم تنفيذ شيء معين

مثال :-

```
$f = 5 ;

if ! ($f == 6 ) { echo " Hi " ; }
```

تلاحظ بأنه سيطبع Hi لأن الشرط منفي .

ملاحظة/ يمكن الجمع بين المعاملات المنطقية (**and , or , xor , !**)

الفصل الرابع

(المصفوفات Arrays)

يتم إنشاء المصفوفة بشكل مشابه لإنشاء المتغير حيث يتم وضع العلامة دولار ($) وبعدها اسم المصفوفة مثل شروط اسم المتغير من حروف وأرقام أو أحدهما وبعدها نضع المساواة وبعدها نضع كلمة array وهذه الكلمة توضع لتعريف المحلل على إنها مصفوفة وبعدها نضع القيم بين قوسين .

مثال :-

```
$x = array ("ahmed","ali","salem") ;
```

في هذا المثال تم فتح المصفوفة $x وتم إعطائها القيم (ahmed ali salem) أي إنها تحتوي على ثلاثة قيم ويكون شكل المصفوفة الوهمي كما بالجدول حيث إن كل قيمة في المصفوفة تأخذ رقملمعيناً يبدأ بالصفر وإلى مالا نهاية .

التسلسل في المصفوفة index	القيمة المعطاة value
0	ahmed
1	ali
2	salem

لاحظ أننا يمكن أن نعطي لكل قيمةرقماً تسلسلي معين من اختيارنا كما في هذا المثال :-

```
$x = array ( 4=> "ahmed" ,  9=> "ali" , 2=> "salem") ;
```

ويمكن تغير قيمة أي عنصرمثلاً سنغير اسم ahmed باسم (لمياء)

```
$x [ 4 ] = " لمياء " ;
```

ويمكن إضافة عنصر جديد إلى المصفوفة

```
$x [  ] = " kaled " ;
```

فأنه سيضيف هذا العنصر إلى المصفوفة x وبما أننا لم نعطه رقم تسلسلي فإنه سيأخذ رقم تسلسلي افتراضي يكون أكبر من رقم أكبر تسلسل موجود ,ومثلاً في هذا المثال سوف يأخذ الرقم (10) .

ويمكن أن نكتب بشكل عمودي هكذا :-

```
$x = array ( );

$x[  ] = " ahmed" ;

$x [6] = " ali ";

$x [2] = " salem" ;
```

حيث أولاً عرفنا اسم المصفوفة $x ثم جعلناها فارغة (لا تحتوي على رقم) وبعد ذلك نضع القيم التي نشاء وبالتسلسل الذي نشاء والتي لم نضع لها رقم تسلسلي تأخذ رقماً افتراضياً , ويمكن التعامل بسهولة مع هذه المصفوفة وطباعة أي صف أو أي عنصر نريد .

مثال :-

```
echo "$x[6] <br> $x[2] <br>" ;
```

ملاحظة / يمكن وضع قيمة نصية في مكان التسلسل , مثل :-

```
$x [ "man" ] = "ahmed" ;
```

* إذا أردنا طباعة عناصر مصفوفة تتكون من عدد كبير من العناصر فلا يمكن أن نستخدم **echo** لطباعة كل عنصر على حده فلذلك نحتاج إلى عبارة تكرار ,مثلاً لو كانت لدينا مصفوفة تحتوي على (100) عنصر ولم نقم بتحديد ترتيب العناصر إذن فنحن نعلم بأن الترتيب يبدأ من الرقم صفر إلى الرقم (99) الذي يمثل قيمة العنصر (100) فبهذه الحالة نستطيع طباعة العناصر باستخدام عبارة التكرار (for) .

مثال :-

```
$people = array ("ahmed" , "ali" ,  .... 100 إلى أن نصل إلى العنصر "Lee" ) ;

for ( $x =0 ; $x < 100 ; $x++ ) {

echo "$people [ $x ] <br> " ; }
```

ملاحظة/ في لغة html إن وضعنا الخاصية name وهي خاصية تستخدم في النماذج وهذا الاسم المعطى يتم التعامل معه على إنه متغير في حال أردنا أن نربطه بلغة php , ولكن إذا أعطيناه اسم وبعدها وضعنا قوسين [ ] فارغين سيتم التعامل مع هذه المدخلات على أنها مصفوفة وتضاف تلقائياً .

مثال :-

صفحة html

```
<select name = " x[  ] " multiple >
<option> Ahmed </option>
<option> Ali </option>
</select>
```

صفحة php

```
for ($a = 0 ; $a < 2 ; $a++ ) { echo "$x [ $a ] <br> "; }
```

دوال المصفوفات

- الدالة **print_r**

اسم هذه الدالة مأخوذ من Print Array أي أطبع المصفوفة حيث إن في هذه الدالة يتم طباعة المصفوفة بشكل هرمي أو متشعب .

مثال :-

```
$x =array ("ahmed","ali");
echo ("<pre>");
print_r ($x);
echo("</pre>");
```

وضعنا الوسمين <pre> و </pre> وبينهم الدالة print_r وذلك لكي يكون الناتج بشكل هرمي ويمكن أن نلغي هذين الوسمين ونكتب فقط الدالة لتكون الطباعة بشكل أفقي .

- الدالة **array_merge**

هذه الدالة مسئولة عن دمج مصفوفتين في مصفوفة واحدة جديدة .

مثال :-

```
$x = array ("a","b","c") ;
$y = array ("d","e","f") ;
$z =array_merge ($x ,$y);
print ("<pre>");
print_r ($z);
print ("</pre>");
```

- الدالة **array_reverse**

تعمل هذه الدالة على عكس عناصر المصفوفة .

مثال :-

```
$x = array ("a","b","c") ;
$y = array_reverse ($x) ;
echo ("<pre>");
print_r($y) ;
echo ("</pre>") ;
```

- الدالة **array_pop**

تعمل هذه الدالة على حذف آخر عنصر في المصفوفة .

```
$x = array ("a","b","c") ;

array_pop ($x) ;

print_r ($x) ;
```

هنا في هذا المثال سيتم حذف العنصر c لأنه آخر عنصر ,مثلاً لو أعطينا العنصر c رقم تسلسلي 10 والعنصر b رقم تسلسلي 15 فانهأيضاً سوف يحذف العنصر c بغض النظر عن التسلسل فأنه يحذف آخر عنصر .

- الدالة **array_shift**

تعمل هذه الدالة على حذف أول عنصر بالمصفوفة .

مثال/

```
$x = array ("a","b","c") ;

array_shift ($x) ;

print_r ($x) ;
```

- **الدالة array_push**

تعمل هذه الدالة على إضافة عناصر إلى المصفوفة بشكل سهل وسريع ,خصوصاً إن كنا نريد إضافة عدد كبير من العناصر .

مثال :-

```
$a [ ] = "ali " ;

$a [ ] = "salem" ;

array_push ( $a, ahmed , kalid ) ;
```

لاحظ هنا أضفنا عنصرين إلى المصفوفة , ويمكن إضافة عنصر واحد أو أكثر كما نشاء , لكن أولاً تكتب اسم المصفوفة ثم توضع العناصر , ويمكن أن نضع قيم العناصر بين علامات تنصيص

- **الدالتين implode و explode**

الدالة **implode** تعمل على إضافة قيمة معينة في ما بين قيم المصفوفة , وتحول هذه المصفوفة إلى متغير (قيمة نصية) .

مثال/

```
$array = (“a” , “b” , “c” ) ;

$x = implode( “ - “ , $array ) ;

echo $x ;
```

في هذا المثال تحولت المصفوفة إلى متغير ويمكن التعامل معها وفق هذا الأساس وفي هذا المثال ستكون النتيجة (a - b - c) وكما تلاحظ فقد تم الفصل بين القيم بعلامة – وهي التي أدخلناها في البارامتر الأول للدالة ويمكن إدخال أي علامة أو كلمة .

أما الدالة **explode** تعمل عكس الدالة implode حيث أنها ستحول المتغير إلى مصفوفة بحسب محرف معين يجب أن يكون مضاف في ما بين عناصر المتغير .

مثال :-

```
$t = "a - b - c" ;
$r = explode ( " - " ,  $t ) ;
echo  $r[1]  ;
```

حيث هنا في هذا المثال تم تحويل المتغير $t إلى مصفوفة وما بين كل علامة (-) تكون قيمة من قيم المصفوفة وفي هذا المثال سيطبع القيمة B

- **الدالة count**

تعمل هذه الدالة على حساب عدد عناصر المصفوفة .

مثال :-

```
$c = array ("A" , "B" , "C") ;
$x = count ($c) ;
echo $x ;
```

- **الدالة array_count_values**

عندما نريد أن نتعرف على تكرار قيم في مصفوفة معينة فإننا نستخدم هذه الدالة لتعود لنا بمصفوفة جديدة تحتوي على اسم العنصر وعدد مرات تكرره في المصفوفة .

مثال /

```
$x=array_count_values( $myarray ) ;
foreach( $x  as  $key => $value )
{ print( "$key : $value <br> \n" ); }
```

- **الدالة key**

تعمل هذه الدالة على إيجاد رقم العنصر (تسلسله) في المصفوفة , أي رقم أول عنصر (العنصر النشط) .

مثال :-

```
$a = array ( " ali " , " salem " ) ;
$t = key ( $a ) ;
echo $t ;
```

- **الدالة array_keys**

تستخدم هذه الدالة لمعرفة أرقام العناصر وأسماء العناصر في المصفوفة .

مثال/

```
$x=array_keys( $myarray ) ;
```

- **الدالة current**

تعمل هذه الدالة على إيجاد قيمة أول عنصر في المصفوفة , أي قيمة أول عنصر (العنصر النشط)

مثال :-

```
$a = array ( " ali " , " salem " ) ;
$t = current ( $a ) ;
echo $t ;
```

ملاحظة/ يمكن تنشيط أي عنصر في المصفوفة وذلك عن طريق الدالتين (**next**) و (**prev**) اللتان تعملان على التجول بين عناصر المصفوفة , لاحظ إن العنصر النشطافتراضياً هو العنصر الأول وإن أردنا أن ننشط العنصر الثاني نضع الدالة (**next**) مرة واحدة , وإن أردنا أن ننشط العنصر الثالث نضع الدالة (**next**) مرتين وهكذا نستمر عنصر بعد عنصر , وإن أردنا أن نرجع خطوة للخلف نضع الدالة (prev) مرة واحدة وإن أردنا أن نرجع خطوتين نضع الدالة (**prev**) مرتين وهكذا نرجع خطوة بعد خطوة إذاً يمكننا أن نتقدم خطوة خطوة بقدر ما نشاء أو نرجع خطوة خطوة بقدر ما نشاء .

مثال :-

```
$a = array ( "A", "B" , "C" ) ;
next ($a) ;
next ($a) ;
prev ($a) ;
echo key ($a) . "<br>" ;
echo current ($a) . "<br>" ;
```

وهكذا فنحن نشطنا العنصر الثاني .

- **الدالة (list و each)**

تستطيع باستخدام هذه الدالتين وعن طريق التكرار (**while**) استخراج جميع عناصر المصفوفة

مثال :-

```
$a = array ("A" , "B" ) ;
while (list ($e , $r) = each ($a) ) {
echo " <br> $e <br> $r " ; }
```

سأشرح المثال الآن :-

while هي أداة تكرار نعرفهامسبقاً حيث كتبنا في شرطها الدالتين **list** و **each** وسوف يتوقف الشرط (يصبح غير صحيح) بعد انتهاء عناصر المصفوفة حيث نضعأولاً الدالة **list** وبين قوسين كتبنا متغيرين ونستطيع تحديد أسمائهم كما نشاء حيث إن العنصر الأول يشير إلى رقم العنصر في المصفوفة ويشير المتغير الثاني إلى قيمة هذا العنصر ويمكن أن نحذف أحد هذين المتغيرين إن لم نحتاج إلى رقم العنصر أو لم نحتاج إلى معرفة قيمة العنصر مع ملاحظة عدم حذف الفارزة الموجودة بين المتغيرين إذا حذفنا المتغير الأول (متغير التسلسل) ولكننا نحذف الفارزة إذا حذفنا المتغير الثاني (متغير القيم) , بعد ذلك أغلقنا قوس الدالة **list** وجعلناها تساوي الدالة **echo** ونضع بين قوسي الدالة اسم المصفوفة المعرفةمسبقاً والتي نريد من أداة التكرار **while** أن تدور بقدر عناصر هذه المصفوفة ثم نغلق قوس الشرط للأداة **while** ونفتح قوس تنفيذ الأمر للأداة **while** إذا تحقق الشرط (وهو الدوران بعدد عناصر المصفوفة) حيث إننا اخترنا في هذا المثال أن نطبع رقم العناصر مع قيمها ويمكن تنفيذ أمور أخرى . بحث إننا نستطيعمثلاً أن ندخل قيم وأرقام عناصر المصفوفة في جدول في لغة html أو أن نفعل أشياء أخرى .

- **أداة تكرار المصفوفات (foreach)**

هذا التكرار من الأشياء الجيدة في php وهو يساعدك على معرفة عناصر مصفوفة معينة أو طباعة محتوياتها , والصيغة العامة لها هي :-

foreach (اسم المصفوفة As اسم متغير1 => اسم متغير2) { code }

حيث إن (اسم متغير 1) يمثل التسلسل للمصفوفة و (اسم متغير 2) يمثل قيم المصفوفة .

مثال :-

```
$t = array (a=> "ahmed" , b=> "basim" , c=> "car") ;
foreach ( $t As $x => $y ) { echo ( $x . "-----" . $y ) ; }
```

- **الدالة is_array( ) ;**

تقوم هذه الدالة بالتحقق من الوسيط الممرر لها هل هو مصفوفة او لا وذلك بإعادة القيمة true او false والصيغة العامة لها هي

```
is_array($array);
```

- **الدالة in_array**

تعود بالقيمة true إذا كانت القيمة موجودة في داخل المصفوفة .

مثال/

```
$test=array( "a" , "b" , "c" ) ;
if( in_array( "b" , $test ) )
print "found b" ;
```

- **الدالة array_unique**

تقوم هذه الدالة بإزالة أي قيمة تتكرر في المصفوفة , حيث تعيد مصفوفة بدون أي عناصر مكررة والصيغة العامة لها هي :-

```
array_unique($array);
```

- **الدالة range**

تستخدم هذه الدالة مع المصفوفات الرقمية لإنشاء أرقام متسلسلة

مثال/

```
$x = range (1 , 50 , 1) ;
```

حيث هنا أنشأنا مصفوفة اسمها $x ووضعنا الدالة range ووضعناأولاً في القوس الرقم 1 الذي يمثل الرقم الذي سيبدأ منه العد وبعده الرقم 50 وهو يمثل آخر رقم وبعده الرقم 1 وهو يمثل مقدار الزيادة التي ستجري حيث سيتم إضافة الرقم واحد في كل مرة

- **extract**

تستخدم هذه الدالة مع المصفوفات التي نضيف لها اسماء بدل ترقيم الفهرسة الأعتيادية وفائدتها هي لأختصار الوقت حيث أنها ستمكننا من التعامل مع الاسم الذي نضيفه إلى ترقيم قيمة المصفوفة كمتغير .

مثال /

```
$arr = ( “x” => “value1”, “y” => “value2” , “z” => “value3” ) ;

extract( $arr ) ;

echo  $x ;
```

وفي هذا المثال سيطبع لنا القيمة value1

- **الدالة var_dump**

وهي تستخدم لطباعة المصفوفة ولكن بشكل مرتب , مثال/

```
var_dump($x) ;
```

- الدالة **array_pad**

تقوم هذه الدالة بجعل قيم المصفوفة تساويعدداً معينا وتضع بدل القيم الناقصة القيه الافتراضية التي ترسل لها .

مثال/

```
$my=array( "a" , "b" , "c" ) ;
$x=array_pad( $my , 5 , zzz ) ;
print( "<pre>" );
print_r( $x ) ;
print( "</pre>\n" ) ;
```

القيم التي ستضاف مكان القيم الناقصة

عدد عناصر المصفوفة

- الدالة **array_slice**

تقوم هذه الدالة بنسخ مجموعة قيم من مصفوفة على شكل مصفوفة جديدة ويتم الحصول على القيم عن طريق تحديد بداية النسخ وعدد القيم التي تنسخ .

مثال/

```
$x=array_ slice( $myarray , 2 , 3 ) ;
```

حيث أن الرقم 2 هنا يعبر عن تسلسل أول عنصر سينسخ والرقم 3 يمثل عدد العناصر التي ستنسخ بعد العنصر الأول (الذي حددناه)

- الدالة **array_value**

تعود هذه الدالة بجميع قيم المصفوفة في مصفوفة جديدة .

دوال الحركة لمؤشر قراءة المصفوفة

الدالة	عملها
Reset	تصغير المؤشر ووضعه في بداية المصفوفة .
next	التوجه للعنصر التالي .
Rrev	التوجه للعنصر السابق .
current	قيمة المتغير الحالي .
End	وضع المؤشر عند آخر عنصر .

فرز المصفوفات

- **الدالة sort ( );**

هذه الدالة تأخذ محتويات المصفوفة ومن ثم تفرز عناصرها هجائياً اعتماداً على الأحرف الكبير أولاً ثم الصغيرة , تتطلب هذه الدالة فقط اسم المصفوفة , لاحظ هذا المثال :-

```
$n = array ("Ahmed" , "ali" , " Basem" , "kalid") ;
sort ($n) ;
while ( list ($x , $y) = each ( $n) ) {
echo "<br> $x <br> $y " ; }
```

- **الدالة asort ( );**

هذه الدالة تعمل نفس عمل الدالة **sort** ولكن الفرق بينهما هو إن الدالة **sort** تستبدل الحروف بأرقم في الفهرسة أما الدالة **asort** تضع الحروف كما هي وتفرزها كما تفعل الدالة **sort** , والصيغة العامة لها هي :-

```
asort ( اسم المصفوفة ) ;
```

- **الدالتين rsort و arsort**

تعمل الدالة **rsort** نفس عمل الدالة **sort** ولكن بشكل عكسي , وتعمل الدالة **arsort** نفس عمل الدالة **asort** ولكن بشكل عكسي , والصيغة العامة لهذه الدالتين هي :-

```
rsort ( اسم المصفوفة ) ;
arsort ( اسم المصفوفة ) ;
```

ملاحظة/ يمكن استعمال دوال الفرز مع الحروف العربية (إذا كان السيرفر يدعم اللغة العربية) .

- **الدالة ksort**

تعمل هذه الدالةأيضاً على فرز المصفوفات لكن بالاعتماد على رقم تسلسل العنصر في المصفوفة وليس بالاعتماد على قيمة العنصر , والصيغة العامة لها هي :-

```
ksort ( اسم المصفوفة ) ;
```

تداخل المصفوفات

يمكن صناعة مصفوفة داخل مصفوفة أو أكثر من مصفوفة أي يمكن أن تتداخل المصفوفات .

مثال :-

```
$t = array (1=> array ("A" , 23) , 2=> array ("B" , 25)) ;
while ( list($x) = each ($t) { echo ("<br> $x <br> " ) ;
while ( list( , $y) = each ($t[$x]) { echo ("$y") ; } }
```

* ولطباعة قيمة معينة من المصفوفة الداخلية نكتب قيم المصفوفة وبعد ذلك رقم المصفوفة .

مثال/

```
$x=array( ) ;
$x[ ]=array ("Hi" , "hello") ;
$x[ ]=array("welcome" , "good") ;
echo $x[ 0 ] [ 1 ] ;
```

هنا سيطبع فقط القيمة hello

الفصل الخامس
(الوقت والتاريخ)

*** time stamp**

وهي الصيغة المشفرة للوقت ويتم استخراجها من خلال الدالة **time** .

مثال :- **echo time ( ) ;**

*** عرض الوقت والتاريخ**

يتم ذلك من خلال الدالة **date** ولكن يجب أن نخبره عن طريقة العرض .

مثال :- **echo date (" j/n/y ") ;**

لتكون النتيجة هكذا :- 25/2/2012

*** التحول من time stamp إلى التاريخ العادي**

يمكن ذلك بطريقتين :-

1- الدالة (date)

مثال :-

```
echo date ( "j/n/y" , 1122350269 ) ;
```

هذا الرقم يمثل الوقت المشفر time stamp

2- الدالة (getdate)

يتم في هذه الدالة تحويل التاريخ والوقت من الصيغة العشوائية time stamp إلى الصيغة العادية للوقت , وتكون المخرجات بشكل مصفوفة .

مثال :-

```
$x = getdate ( 1122350269 ) ;
print ("<pre>") ;
print_r ($x) ;
print ("</pre>") ;
```

* التحول من تاريخ عادي إلى time stamp

يتم التحويل هنا باستخدام الدالة **Mktime** وهي تحتاج بعض المدخلات لاحظ :-

```
Mktime ( سنة , شهر , يوم , ساعة , دقيقة , ثانية );
```

* الدالة gmdate

وهي تستخدم لعرض التاريخ

مثال :-

```
echo gmdate ( m ) ;
echo gmdate ( M ) ;
```

سوف تلاحظ إن النتائج سوف تكون مختلفة على الرغم من استخدام نفس الدالة , لكن تختلف القيمة المعطاة (m) (M) ولغة php تحتجز الكثير من الكلمات والحروف التي تقوم بعمليات مختلفة .

ويمكن استعمال حروف أخرى مع الدالة **gmdate** لاحظ هذا المثال :-

```
echo gmdate (" D , d M Y H : i : S ") ;
echo gmdate (" M D") ;
```

- **الدالة microtime( ) ;**

تستخدم هذه الدالة لإخراج الوقت لكن بالملي ثانية و هي لا تأخذ أي وسيط بين قوسيها .

* لاحظ هذا الجدول الخاص بالرموز :-

الرمز	الوصف	مثال على المخرجات
a	الوقت صباحاً أو مساءا small	am , pm
A	الوقت صباحا أو مساءا capital	AM , PM
d	رقم اليوم في الشهر يبدأ بصفر	من 01 إلى 31
D	اسم اليوم بشكل مختصر	Sun , Mon
F	اسم الشهر بدون اختصار	August
g	رقم الساعة الآن بصيغة 12 ساعة لا يبدأ بصفر	1 - 12
G	رقم الساعة الآن بصيغة 24 ساعة لا يبدأ بصفر	0 - 23
h	رقم الساعة الآن بصيغة 12 ساعة يبدأ بصفر	01 - 12
H	رقم الساعة الآن بصيغة 24 ساعة يبدأ بصفر	00 - 23
i	عدد الدقائق في الساعة	00 - 59
I	تحديد الصباح والمساء على شكل رقمين 0 أو 1	1 صباح و2 مساء
j	رقم اليوم في الشهر ولا يبدأ بصفر	1 - 31
l	اسم اليوم كاملاً بدون اختصار	Friday
L	حالة السنة كبيسة أو لا	1 كبيسة و2 غير كبيسة
m	رقم الشهر في السنة ويبدأ بصفر	01 - 12
M	اسم الشهر في السنة مختصر	Jan
n	الشهر في السنة على شكل رقم ولا يبدأ بصفر	1 - 12
s	الثواني في الدقيقة على شكل رقمين	00 - 59
S	اسم اليوم مختصر	TH , ST , ND
t	عدد الأيام في الشهر	28 - 31
U	Time stamp	1222352049
w	اليوم من الأسبوع على شكل رقم	0 - 7
y	السنة على شكل رقم مكون من رقمين	88,89,90
Y	السنة على شكل رقم مكون من أربع أرقام	2012,2013
z	اليوم في السنة على شكل أرقام	0 - 365

الفصل السادس

(الدوال الرياضية والتعامل مع الأرقام)

الدالة	شرحها	مثال عليها
bcadd	تتيح هذه الدالة جمع رقمين مهما كان نوعهما كما تتيح لك تحديد عدد الأرقام بعد الفاصلة العشرية التي تدخل ضمن هذه العملية , حيث أن الرقم الأول يمثل العدد الأول الذي سيتم جمعه والرقم الثاني يمثل العدد الثاني الذي سيتم جمعه مع الأول والرقم الثالث يمثل عدد الأرقام التي ستحسب بعد الفارزة .	bcadd("1.5","9.87", 2) ;
bccomp	تقوم هذه الدالة بالمقارنة بين رقمين لتعود لك بنتيجة المقارنة فإذا كان الرقمان متساويان تعود بالقيمة صفر أما إذا كان الرقم الموجود على اليسار أكبر تعود بالقيمة 1 وإذا كان الرقم الموجود على اليمين أكبر تعود بالقيمة 1- مع ملاحظة أن هذه الدالة لا تأخذ بعين الاعتبار الرقم بعد الفارزة إلا إذا أعطيناها رقم ثالث يشير إلى الأرقام .	bccomp("1","1.122", 2);
bcsqrt	تقو هذه الدالة بإعادة الجذر التربيعي للرقم الذي سوف نرسله لهاوأيضاً نستطيع أن نحدد الأرقام بعد الفاصلة العشرية التي سنعرضها .	bcsqrt("17" , 3) ;
sqrt	تعود بالجذر التربيعي للرقم المرسل لها .	sqrt(9.9) ;
abs	تعود بالقيمة الحقيقية للرقم	abs(-13) ;
max	تعود بأكبر قيمة من بين القيم المرسلة إليها سواء كانت أرقام او نصوص .	max("arab" , 15 , 33) ;
min	نفس الدالة السابقة لكنها تعود بأصغر قيمة من القيم المرسلة إليها .	min("arab" , 15 , 33) ;
ceil	تقرب الرقم إلى أكبر رقم صحيح لاحق .	ceil(13.01) ;
floor	تستخدم هذه الدالة لتقريب الرقم إلى أقرب قيمة دنيا	floor(13.02) ;
log	للحصول على لوغاريتم العدد .	log(20.13) ;

الفصل السابع
(تعريف واستدعاء الدوال)

يمكن ترتيب دالة بالشكل الذي نريد وبعد ذلك نستخدمها (نستدعيها) في أي مكان وذلك من خلال (**function**) , والصيغة العامة لها هي :-

function اسم الدالة (المتغيرات) { وضيفة الدالة (الكود) }

مثال:-

```
function bob ($x) {
$x = $x + 100 ;
return $x ; }
```

حيث أنشأنا في هذا المثال دالة اسمها bob وفيها متغير وحيد اسمه $x , لاحظ إننا في النهاية وضعنا (**return**) حيث إن وضيفتها هي إخبار الدالة بأن وضيفتها انتهتوأيضاً لكي تخبر php ما هي القيمة التي سيتم اتخاذها إذا كنا نستخدم أكثر من قيمة , لاحظ في مثالنا هذا استخدمنا متغير واحد ولكن إن كان هناك أكثر من متغير فيجب أن نعرف php أيهما هو المتغير الرئيسي .

مثال :-

```
function bob ($a , $b) {
$a = $a + 100 ;
$b = $b + 50 ;
return $a ; }
echo bob (10 , 7) ;
```

في هذا المثال استدعينا وطبعنا الدالة حيث وضعنلمباشرتاً رقماً أو نستطيع وضع متغير معرف مسبقاً مكان الرقم , ولاحظ أن الرقم الأول يمثل البارامتر الأول والثاني يمثل البارامتر الثاني .

مثال :-

```
function bob ($a) {
$a = $a - 1 ;
echo $a ;
return $a ; }
$a = 100 ;
bob ($a) ;
```

نلاحظ في هذا المثال عدة ملاحظات :-

1- استخدمنا اسم المتغير ($a) خارج نطاق تعريف الدالة وكأنه متغير جديد لأن php لا تستطيع التعرف على هذا الاسم فنستطيع استخدامه كأي اسم .

2- جعلنا بين قوسي الدالة bob اسم متغير الذي عرفناه على إنه يساوي 100 .

3- استخدمنا (استدعينا) الدالة bob ومن دون أن ندخلها في دالة الطباعة (**echo**) فأنها تطبع النتيجة وذلك لأننا في داخل تعريف الدالة أعطينا أيعاز الطباعة في أي وقت يتم فيه استدعاء الدالة.

ملاحظة/ لا فرق في أن نعرف الدالةأولاً ومن ثم نستدعيها أو أن نقم باستدعائها وبعد ذلك نعرفها.

* يمكن أن نعطي للدالة return القيمة true وهذا يعني أن يتم إعادة تحميل الصفحة مرة أخرى أما إذا أعطيناها القيمة false فهذا يعني التوقف عن إعادة تحميل الصفحة .

* يمكن إعطاء قيمة افتراضية للمتغير ليتم تنفيذها إذا لم تعطى له قيمة ولا تنفذ إذا أعطيت لها قيمة

مثال/

```
<?php
function ex($a=5){
$a = $a + 100 ; }
echo ex();
echo ex( 2 );
```

ملاحظة/ يمكن جعل المتغيرات المحلية (كما قلنلسابقاً يمكن استخدام نفس اسم المتغير المستخدم داخل تعريف الدالة خارجها وذلك لأنها متغيرات محلية) متغيرات عامة (أي تحتفظ بقيمتها ويمكن استخدامها في أي مكان لتعطي نفس النتيجة) وذلك بإضافة (**global**) قبل اسم المتغير داخل التعريف ليكون هذا المتغير عام في كل اللغة .

مثال :-

```
function  ala( ) {
global $x ;
$x = "program" ;
return  $x ; }
echo $x ;
```

كذلك يمكننا أن نعرف المتغير المستخدم داخل الدالة خارجها وذلك باستخدام المصفوفة ($GLOBALS) ونكتب بين قوسيها اسم المتغير ولاحظ أنه بدون العلامة $.

مثال/

```
function  test( ) {
return  $GLOBALS[ " var " ] ;
}
$var = "value" ;
echo test( ) ;
```

ملاحظة/ عند تعريف متغير خارج الدالة function فإنه لا يمكننا استخدامه داخلها لأنها سوف لن تتعرف عليه ولتعريف هذا المتغير للدالة يجب أن نسبقه بـ (global) في داخل الدالة .

مثال/

```
$var  = ' value ' ;

function  test( ) {
global  $var ;
echo $var ;
 }
test( $var ) ;
```

تداخل الدوال

يمكن أن نضع تعريف دالة داخل تعريف دالة أخرى .

مثال :-

```
function bob($x) { $x = $x - 1 ;
function gbg($y) { $y = $y + 4y ;
return $y; }
$x = gbg($x) ;
return $x ; }
echo bob(15) ;
```

الفصل الثامن

(استخدام JSON لتخزين وجلب البيانات)

التنسيق JSON :

JSON وهي اختصار لـ javaScript Object Notation وهي طريقة في لغة javaScript للتعامل مع البيانات وتم انتشارها ودعمها في أغلب لغات البرمجة الأخرى لسهولة وديناميكية التعامل مع هذه الطريقة ويمكن لنا استخدام هذه الصيغة كبديل أمثل لنقل البياناتبدلاً من استخدام ملفات XML وأيضاً تستخدم هذه التقنية في جلب البيانات .

ولإنشاء JSON يتم حفظ البيانات في تنسيق JSON على شكل كائن وتوضع العناصر بين الأقواس { } او على شكل مصفوفة وتوضع عناصر المصفوفة بين القوسين [ ] , القيم التي يتم حفظها داخل الكائن او المصفوفة هي أعداد صحيحة وأعداد كسرية وسلاسل نصية وقيم منطقية وكائنات أخرى او مصفوفات أخرى ويمكن الجمع بين جميع هذه الأنواع داخل كائن واحد ويتم إسناد القيم للعناصر باستخدام النقطتين (:) ويتم الفصل بين العناصر باستخدام الفارزة (,) , وكمثال للقيم داخل كائن لاحظ هذا الكود :-

```
{"var1":10,"var2":true,"var3":null,"var4":"value","var5":12.55}
```

ملاحظة/ يجب أن يكون اسم العنصر بين علامتي اقتباس لأن بعض لغات البرمجة لا تقبل اسم العنصر بدونهاوأيضاً يجب وضع السلسلة النصية بين علامات اقتباس , وكمثال للقيم داخل مصفوفة لاحظ هذا الكود :-

```
[10,20.25,"value",null,true]
```

كما يمكن الجمع بين الاثنينمعاً كأن يحتوي الكائن على مصفوفات او تحتوي المصفوفة على كائنات , لاحظ هذا المثال :-

```
{"var1":10,"var2":[10,20,30]}
```

ولاحظ هذا المثالأيضاً :-

```
[10,20,{"var1":"value1","var2":900},"value2"]
```

- الدالة **json_encode**

تستخدم هذه الدالة لتحويل البيانات إلى صيغة JSON

مثال/

```
<?php
$data['var1'] = 10;
$data['var2'] = 20.13;
$data['var3'] = null;
$data['var4'] = true;
$data['var5'] = 'value';
echo json_encode($data);
?>
```

وستكون النتيجة بهذا الشكل

```
{"var1":10,"var2":20.13,"var3":null,"var4":true,"var5":"value"}
```

مثال/

```
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
echo json_encode($arr);
?>
```

- الدالة **json_decode**

تستخدم هذه الدالة لتحويل صيغة JSON إلى كائنات ومصفوفات

مثال/

```
<?php
$json = ' { "var1":10 , "var2":true , "var3":null , "var4":"value" , "var5":12.55 } ' ;
$data1 = json_decode($json);
$data2 = json_decode($json , true);
// للوصول للعناصر من خلال الكائن
echo $data1->var4;
echo "<br>";
// للوصول للعناصر من خلال المصفوفة
echo $data2['var4'];
?>
```

ستكون النتيجة بهذا الشكل :-

value
value

الفصل التاسع
(رفع الملفات إلى الخادم)

لرفع ملف من الموقع إلى الخادم يجبأولاً وعند إنشاء حقل النموذج في الـ HTML أن نحدد الطريقة post لإرسال البيانات بالإضافة إلى وضع ترميز النموذج "multipart/form-data" بدلاً من الترميز الافتراضي "application/x-www-form-urlencoded" .

مثال/ لإنشاء نموذج بسيط لرفع البيانات

```
<form action="file_upload.php"  method="post"  enctype="multipart/form-data">
<input type="file" name="file1">
<input type="submit">
</form>
```

- **المصفوفة $_FILES**

تُخزن هذه المصفوفة معلومات عن الملف أو الملفات التي تم رفعها إلى الخادم . و بشكل عام هذه المصفوفة ثنائية البعد حيث يُعبر البعد الأول عن اسم حقل الملفالمُحدد في حقل الملف في نموذج HTML أما البعد الثاني , فيوفر معلومات عن اسم الملف أو حجمه أو نوعه أو رسالة الخطأ في حال وجودها .

* الجدول التالي يوضح قيم المصفوفة السابقة , وتم اعتماد file1 كاسم لحقل الملف (للتوضيح فقط)

القيمة	الشرح
$_FILES['file1']['name']	يُحدد هذا المتغير اسم الملف الأصلي كما هو في جهاز المستخدم .
$_FILES['file1']['size '] :	كما هو واضح من الاسم , يُحدد هذه القيمة حجم الملف الذي تم رفعه مقدرا بالبايت , لذا قد تحتاج إلى قسمة هذا الرقم على 1024 او 1024^2 للحصول على حجم الملف مقدرا بالكيلوبايت أو الميغابايت على التوالي .

$_FILES['file1']['type']	تحدد هذه القيمة ما يسمى MIME type للملف الذي رفعه , فمثلاً تكون قيمة MIME type لصورة من صيغة png هي image/png او لملف pdf هي application/pdf , ملف مضغوط من نوع zip هي application/zip
$_FILES['file1']['tmp_name']	تُحدد هذه القيمة اسم الملف المؤقت المُخزن على الخادم , وسيتم استخدام هذه القيمة كثيرا عند استدعاء الدوال الخاصة برفع الملفات كما سنرى لاحقاً .
$_FILES['file1']['error']	لا يمكن بداً ضمان سير عملية رفع ملف على الخادم بشكل صحيح , وبعض الحيان تكون هنالك مشكلة في رفع الملفات و من المُفيد معرفتها و تبليغ المستخدم عن سبب الخطأ , حيث تُعيد القيمة رقم رسالة الخطأ أو الثابت الموافق لها .

مثال /

```
<?php
if($_POST['sub']){
echo $_FILES['NAME']['name'];
}
?>
<form method="post" enctype="multipart/form-data">
<input type="file" name="NAME">
<input type="submit" name="sub">
</form>
```

كما تمت الإشارة إليه في الجدول السابق فإن القيمة $_FILES['file1']['error'] تعيد ثابت او رقم يدل على الخطأ الذي حدث أثناء الرفع , والجدول التالي يوضح أبرز هذه القيم

القيمة او الرقم	الشرح
UPLOAD_ERR_OK , 0	يعيد المتغير $_FILES['file1']['error'] هذه القيمة عندما تتم عملية رفع الملفات بنجاح دون أي أخطاء
UPLOAD_ERR_INI_SIZE, 1	يعيد المتغير $_FILES['file1']['error'] هذه القيمة عندما يتم رفع ملف حجمه يتجاوز الحجم المسموح به المحدد بالراية upload_max_filesize الموجودة في الملف php.ini .

UPLOAD_ERR_PARTIAL , 3	**يتم إعادة هذه القيمة عند رفع جزء من الملف وعدم التمكن من رفعهكاملاً كحدوث مشكلة في الشبكة .**
UPLOAD_ERR_NO_FILE , 4	يتم إعادة هذه القيمة عندما يقوم المستخدم من إرسال نموذج HTML دون تحديد ملف لكي يتم رفعه .

وبالطبع يمكن التحقق من رسالة الخطأ أما باستخدام الأرقام أو بمساواتهم بالثوابت السابقة .

مثال/ يقوم بطباعة حالة رفع ملف

```
<?php
switch ($_FILES['file1']['error'])
{
case UPLOAD_ERR_OK:
echo "File uploaded succesfuly";
break;
case UPLOAD_ERR_INI_SIZE:
echo "Uploaded File is too big";
break;
case UPLOAD_ERR_PARTIAL :
echo "File is not completely uploaded";
break;
case UPLOAD_ERR_NO_FILE :
echo "No File was Selected";
break;
default:
echo "UnKnown Error";
break;
}
?>
```

دوال رفع الملفات

- الدالة is_uploaded_file

تستخدم هذه الدالة للتأكد من رفع الملف وتعيد القيمة true في حال تم الرفع و false فيما خلاف ذلك , وتأخذ وسيط وحيد بين قوسيها وهو الاسم المؤقت للملف , وتكون الصيغة العامة ها بهذا الشكل

```
is_uploaded_file($filename);
```

- الدالة move_uploaded_file

تستخدم هذه الدالة لنقل ملف تم رفعه إلى مجلد معين وتقبل هذه الدالة وسيطين : الأول هو اسم الملف المؤقت والثاني هو المسار الهدف الذي سيتم نقل الملف إليه , والصيغة العامة لها هي

```
move_uploaded_file($tmp_name, $distination);
```

مثال/

```
<?php
if($_POST['sub']){
$name_file  = $_FILES['NAME']['name'];
$temp_file  =  $_FILES['NAME'][ 'tmp_name'];
$folder  = "Myfolder";
move_uploaded_file( $temp_file  ,  $folder."/".$name_file );
}
?>
<form  method="post"  enctype="multipart/form-data">
<input type="file" name="NAME">
<input type="submit" name="sub">
</form>
```

ملاحظة/ اذا حاولنا رفع ملف الى مجلد وكان نفس الملف موجود (نفس الاسم والامتداد)مسبقاً في المجلد فأنه سوف يتم استبدال الملف القديم بالملف الجديد .

رفع عدة ملفات

يمكن رفع عدة ملفاتسوياً وبهذه الحالة ستكون المصفوفة $_FILES ثلاثية الأبعاد حيث سيكون البعد الثالث هو رقم حقل الملف ويبدأ العد - كالعادة - من القيمة صفر حيث يكون اسم الملف الأصلي لأول حقل ملف هو $_FILES['file']['name'][0]: وللملف الثاني $_FILES['file']['name'][1]: ... الخ , ويكون نموذج HTML كالتالي :

```
<form  action="file_upload.php" method="post"
enctype="multipart/form-data">
<input type="file" name="file[ ]"> <br>
<input type="file" name="file[ ]"> <br>
<input type="file" name="file[ ]"> <br>
<input type="submit">
</form>
```

الفصل العاشر
(دوال منوعة)

- **الدالة rand**

وظيفة هذه الدالة إنها تعمل أرقام عشوائية

مثال :-

```
$x = rand ( 1 , 10 ) ;
```

حيث إنها ستعمل على اختيار رقم عشوائي من الواحد إلى العشرة (بحسب ما نريد ويمكن وضع أرقام أخرى بين القوسين لكي تختار الدالة الرقم من بين هذين الرقمين) ووضعه في المتغير (x) .

- **getenv("REMOTE_ADDR”);**

تتستخدم هذه الدالة للحصول على ip المستخم , لاحظ في هذا المثال سنقوم بطباعة ip الزائر للموقع , مثال /

```
$x = getenv("REMOTE_ADDR”);
Echo $x ;
```

- **$_SERVER['REMOTE_ADDR'] ;**

تستخدم هذه الدالة للحصول على ip الزائر وهي تماماً نفس الدالة السابقة .

مثال/

```
$ x = $_SERVER['REMOTE_ADDR'];
echo $x ;
```

- **الدالة get_magic_quotes_gpc( );**

عند ادخال المستخدم في الحقل النصي علامة تنصيص مفردة او مزدوجة فأن معالج اللغة (السيرفر)سيضع قبل كل علامة تنصيص علامة سلاش خلفية وهذا الأمر مهم جداً للحماية لكن في بعض السيرفرات لا تكون عملية الإضافة هذه مفعلة لذلك نحتاج الى دالة الحماية addslashes لأضافة علامة السلاش (وسيتم شرح الدالة addslashes في الفصول القادمة) , وهنا تظهر فائدة الدالة get_magic_quotes_gpc حيث أنها ستعيد القيمة 1 او true اذا كانت عملية اضافة السلاش مفعلة من قبل السيرفر وتعيد القيمة 0 او false اذا لم تكن مفعلة .

- **wordwrap**

تستخدم هذه الدالة لاضافة حرف معين بعد عدد معين من الحروف وهي تأخذ أربع قيم الأولى تمثل المتغير الذي يحمل الكلمات والقيمة الثانية تمثل عدد الحروف التي سيبدأ بعدها أضافة الحرف الجديد والقيمة الثالثة تمثل الحرف الجديد والقيمة الرابعه هي true

مثال/

```
$x = “sssssssssssssssssssssssssssss”;

$y = wordwrap( $x , 3 , ‘  ‘, true);
```

حيث هنا في هذا المثال أضفنا فاصلة بعد كل ثلاثة أحرف .

- **mail**

تستخدم هذه الدالة لإرسال رسالة إلى بريد اليكتروني معين , لاحظ أن هذه الدالة لا تعمل إذا كنا نستخدن السيرفر الشخصي , والصيغة العامة للدالة هي :-

```
mail( عنوان البريد الذي سترسل إليه , ’الأستفسار’ , ’نص الرسالة’ ) ;
```

- **الدالة header**

هذه الدالة تستخدم لإرسال بيانات من السيرفر الى متصفح الزائر ويمكن ان تكون هذه البيانات هي عبارة عن صفحة جديدة وبهذا سينقل الزائر مباشرتاً الى هذا الصفحة او يمكن ان تكون ابيانات المرسله هي عبارة عن صور او نصوص او اشياء أخرى وبهذه الحالة سيعرضها على متصفح الزائر ,ويجب ان تنتبه الى أمر مهم جداً وهو انه لا يمكن استخدام أي مخرجات قبل هذه الدالة أي يجب أن تكون هذه الدالة في بداية الصفحة والمقصود بالمخرجات هي مثلاً وسوم html او الدالة echo او الدالة print , الدالة header تتكون من جزئين الأول هو العملية والثاني هو القيمة التي تأخذها هذه العملية .

العمليات هي :-

1- Location وهي مسئوله عما اذا كنا نريد ان نرسل (نحول) الزائر الى صفحة جديدة ,وتحدث هذه العملية بشكل سريع جداً .
2- Content-Type وهذه العملية مسئولة عما اذا كنا نريد ان نرسل الى الزائر ملفات قد تكون صور او نصوص او ... إلخ .

أما القيم فهي :-

1- اذا كانت العمليه هي Location فيجب أن تكون القيمة هي عنوان الصفحة او رابط الصفحة .
2- اذا كانت العملية Content-Type فيجب أولاً أن نحدد نوع الملف الذي نود ارساله فإذا كان صورة نكتب images ثم سلاش /ثم امتداد الصورة مثلاً (gif , jpg , ...إلخ) , اما اذا كان الملف نص نكتب text ثم سلاش / ثم نحدد نوع النص مثلاً (html , css , javascript , ... إلخ) , اما اذا كان نوع الملف تطبيق (مستند) نكتب application ثم سلاش /ثم نكتب الأمتداد مثلاً (zip , doc , pdf , ... إلخ) , وتسمى هذه القيم بالـ MIME type .

لاحظ بأنه نفصل بين العملية والقيمة بالعلامة (:) لكن يجب أن تكون هذه العلامة ملاصقة للعملية أي أن لا يكون هناك فراغ (مسافة) بينها وبين العملية ثم نضع فاصلة ثم القيمة .

مثال/لنقل الزائر مباشرتاً الى صفحة أخرى والتي اسمها page2

```
header( "Location:  page2.html" );
```

مثال/ لعرض مستند pdf

```
header( "Content-Type:  application/pdf" );
```

دوال التعامل مع مخرجات البيانات

لاحظ بأن هذه الدوال تستخدم عندما يكون هناك مشكلة في المخرجات خصوصاً عند استخدام الدالة header او عند استخدام الكعكات cookie او عند استخدام الجلسات session , وكل ما سنقوله في هذه الدوال عن الدالة header فهو ينطبق على الكعكات وعلى الجلسات أيضاً (والتي سنأخذها في فصول قادمة) .

- **الدالة ob_start( )**

كما قلنا سابقاً اذا استخدمنا عمليات اخراج للبيانات قبل الدالة header فأن المتصفح سيعرض لنا خطأ لأني لا يدري ماذا سيعرض هل يعرض المخرجات التي تسبق الدالة header ام يعرض مخرجات الدالة نفسها , لكن باستخدام الدالة ob_start( ) فأنه لن يظهر لنا أي خطأ وسيعرض مخرجات الدالة header , لكن لاحظ بأنه يجب أن تكون المخرجات التي تسبق الدالة heared والدالة نفسها تكتب بعد ان نكتب ob_start( ) .

- **الدالة ob_end_flush( ) والدالة ob_flush( )**

نستخدم أحد هذه الدوال في نهاية الكود والفرق بينهم هو ان الدالة ob_end_flush( ) تقوم بحذف الملفات بعد ارسالها للزائر اما الدالة ob_flush( ) فأنها لن تحذف الملفات بعد إرسالها للزائر .

مثال/

```
ob_start( );
echo "<html></html>";
header( "Location:  page2.html" );
print "Welcome Ahmed";
ob_end_flush( );
```

- **الدالة ob_get_contents( )**

تستخدم هذه الدالة لتجميع البيانات (المخرجات) وتعرضها دفعه واحدة .

مثال/

```
ob_start( );
print "Welcome Ahmed";
header( "Location:  page2.html" );
$x = ob_get_contents( );
ob_end_flush( );
```

- **الدالة printf**

وهي دالة إخراج تستخدم لدمج النصوص من المتغيرات الديناميكية

مثال/

```
printf ("hello" , 5) ;
```

- **الدالة sprint**

وهي دالة إخراج ولكن يجب إسنادها إلى متغيرأولاً

مثال/

```
$x=sprint ("hello" , 5 )
```

- **الدالة $_POST[ ]**

تستخدم هذه الدالة لاستخراج القيم من النماذج بتمرير اسم حقل الإدخال من النموذج عليها , وتجلب القيم على شكل مصفوفة .

مثال/

```
$_POST[ 'user' ] ;
```

- **الدالة $_GET[ ]**

تستخدم هذه الدالة لاستخراج القيم من روابط الصفحة بتمرير اسم المتغير في الرابط إلى الدالة , وتجلب القيم على شكل مصفوفة .

مثال/ على فرض أن رابط الصفحة هو التالي www.google.com/index.php?id=12230

```
$_GET[ 'id' ] ;
```

- الدالة $_REQUEST[ ]

هذه الدالة تعمل عمل الدالتين السابقتين $_GET , $_POST حيث اذا انها ستستقبل البيانات سواء كانت مرسلة بالطريقة post او بالطريقة get , مثال /

```
$_ REQUEST[ 'id' ] ;
```

- الدالة empty

وهي تتحقق من أن نموذج الإدخال غير فارغ ولاحظ أنها تعيد القيمة true إذا كان مربع الإدخال فارغ

```
if (empty ($_POST[ 'username' ] ) ) {
echo "please enter a username " }
```

- الدالة strlen( ) ;

تفيد هذه الدالة في معرفة عدد الأحرف التي يدخلها المستخدم .

مثال/ لمنع المستخدم من إدخال نصوص أكثر من 30 حرف

```
$user=$_POST[ 'user' ] ;
if (strlen ( $user ) > 30 ) {
echo "that is too much" ;
die ( "Erorr " ) ; }
```

* ويمكن استخدام هذه الدالة لجلب عدد أحرف النصوص التي يدخلها المستخدم

مثال /

```
$x = 'This is a string';
echo  strlen($x);
```

- strcmp

تستخدم هذه الدالة لمعرفة تطابق قيمتين نصيتين

مثال/

```
$var1 = “Ahmed”;
$var2 = “Ahmed”;
if( strcmp($var1,$var2) == 0 ){
echo “Yes”;
} else {
echo “No”; }
```

ستكون القيمة الناتجة Yes

- الدالة **die( ) ;**

وهي تقوم بعرض رسالة خطأ وما بين الأقواس يكون محتوى الرسالة وهي تخرج من الكود (مثل exit) , لاحظ المثال في الأعلى .

- الدالة **exit( );**

وهي تستخدم للخروج من البرنامج (ايقاف عملية قراءة الكود من قبل المتصفح)

- الدالة **strtolower**

تقوم هذه الدالة بتحويل حالة جميع الحرف الانكليزية إلى أحرف صغيرة , وتفيد هذه الدالة على سبيل المثال عند تسجيل المستخدم في الموقع حيث نقوم بجعل جميع أحرفالمُعرف صغيرة حتى لا يكون لدينامستخدمَين بنفس المُعرف , تقبل هذه الدالة وسيطاًوحيداً هو السلسة النصية وتُعيد سلسلة نصية يكون فيها جميع الحرف بالحالة الصغيرة .

مثال/

```
$x = 'This Is A sTrIng 123';
echo strtolower($x);
```

ملاحظة/ الحروف التي بالغة العربية لا تتأثر بهذه الدالة .

- **الدالة strtoupper**

هي تماماً نفس الدالة السابقة لكن الفرق الوحيد هو أن هذه الدالة تحول جميع الأحرف إلى أحرف كبيرة (عكس عمل الدالة السابقة) .

ملاحظة/ الحروف التي بالغة العربية لا تتأثر بهذه الدالة .

- **الدالة str_replace**

تستخدم هذه الدالة لاستبدال نص معين بنص آخر , والصيغة العامة لها هي

str_replace (مكان النص , النص الجديد , النص القديم)

مثال/

```
$x = 'this is a long string !!';
$new_x = str_replace('long', 'short', $string);
echo $new_x ;
```

- **الدالة str_ireplace**

هذه الدالة هي تماماً نفس الدالة السابقة لكن الفرق الوحيد هو أن هذه الدالة غير حساسة لحالة الأحرف الكبيرة والصغيرة .

- **الدالة phpversion**

تفيد هذه الدالة في معرفة نسخة الـ php (إصدارها) الموجودة لدينا , مثال/

```
echo phpversion( ) ;
```

- **الدالة zend_version**

تفيد هذه الدالة في معرفة اصدار محرك zend الذي تعمل عليه اللغة , مثال /

```
echo zend_version( ) ;
```

- **الدالة php_uname**

تفيد هذه الدالة في معرفة نوع نظام التشغيل , مثال /

```
echo php_uname( ) ;
```

- **الدالة ord**

تستخدم هذه الدالة للحصول على رقم الـ ASCII الخاص بالحرف الممرر لها , مثال /

```
$var = ord('D');
echo $var ;
```

- **الدالة chr**

هذه الدالة هي عكس الدالة السابقة حيث انها تقوم بتحويل الرقم الى ما يعادله من الحروف

مثال /

```
$var = chr(97);
echo $var ;
```

الفصل الحادي عشر
(دوال الأمن والحماية)

- **md5**

تستخدم هذه الدالة لتشفير كلمة المرور التي يدخلها المستخدم .

مثال/

```
$pass = md5( $_POST['password'] );
```

ويمكن كتابتها مرتين او اكثر لزيادة قوة التشفير , لاحظ :-

```
$pass = md5( md5( $_POST['password'] ) );
```

- **الدالة trim( ) ;**

تستخدم هذه الدالة لحذف شيء من بداية ونهاية النص المحدد إذا كان موجود حيث ان هذه الدالة تأخذ بارامترين الأول هو النص المراد تطبيق العملية عليه والثاني هو الشيء المراد حذفه وهذا الباراميتر اختياري لانه في حالة عدم كتابته فأنه سيتم ازالة الفاصلة (الفراغ) من بداية ونهاية النص , وهذه الدالة تستخدم في الغالب مع المدخلات التي يدخلها المستخدم .

مثال /

```
$name = trim($_POST['user'] , 'Ali' ) ;
```

في هذا المثال سيتم ازالة أي حرف من حروف كلمة Ali أين وجد هذا الحرف لوحده أم وجدت كامل الكلمة فأنه سيتم ازالتها لكن انتبه بأنه سيحذف الحروف اذا وجدت في بداية او في نهاية النص المدخل وليس في وسطه .

- الدالة **HtmlspecialChars( ) ;**

إذا قمت بوضع مربع نص وأردت من المستخدم كتابة شيء فيه فأنه يستطيع إدخال أي شيء ولنفرض إنه كتب في مربع النص كالتالي (I am ahmed ...) فسيعمل

المتصفح بعرضها بعد معالجتها كالتالي (I **am ahmed** ...) حيث نلاحظ أن المتصفح يتعامل معها كأنها نصوص html وليس كنص عادي ولكي نعرضها كنص عادي نحتاج إلى هذه الدالة حيث إنها ستعامل كود html كنص عادي وطبيعيتماماً , ونضع هذه الدالة في المتغير في php الذي يأخذ قيمة من مربع النص الموجود في html , لاحظ هذا المثال :-

متغير html

```
$user = HtmlSpecialChars ( $user ) ;

echo $user ;
```

- الدالة **htmlentities( ) ;**

هيتقريباً نفس الدالة السابقة إلا أن الفرق بينهما هو أن هذه الدالة تغير جميع رموز لغة html بخلاف الدالة السابقة التي تمنع بعض من الرموز وليس كلها , لكن بشكل عام يفضل استخدام الدالة السابقة .

ملاحظة// هذه الدالة لا يمكن استخدامها مع اللغة العربية لأنها ستشفر النص ولا يمكن استعادة النص الأصلي .

- الدالة **strip_tags**

تعمل هذه الدالة على انتزاع أي ترويسات او وسوم خاصة بلغات أخرى وتستخدم هذه الدالة للأمان من الهكرز , ويمكن دمجها مع الدالة ; ( )trim لتكون النتيجة هكذا :-

```
$user=$_POST[ 'user' ] ;

strip_tags( trim( $user ) ) ;
```

- **addslashes**

تستخدم هذه الدالة لإضافة علامة السلاش مزدوجة (//) قبل كل علامة تنصيص مفردة ') (وفي العادة تستخدم مع الدالة السابقة .

مثال/

```
addslashes (strip_tags( trim( $user ) )) ;
```

- **الدالة stripslashes( ) ;**

تفيد هذه الدالة في التخلص من علامات السلاش (/) المدخلة في حقول الإدخال .

مثال/

```
$user=$_POST[ 'user' ] ;

stripslashes( $user ) ;
```

ويمكن الأستفادة من هذه الدالة عندما نريد ان نخرج البيانات المخزنة في قاعدة البيانات ولا نريد ان يتم عرض علامات السلاش

- **الدالة escapeshellcmd**

تتخلص هذه الدالة من من اكثر العلامات التي من الممكن ان يكتبها المستخدم مثل & ^ % $ # ; ‘ ” \ * حيث انه سيتم التخلص من هذه العلامات

مثال /

```
$user=$_POST[ 'user' ] ;

escapeshellcmd( $user ) ;
```

- **mysql_real_escape_string**

تستخدم هذه الدالة للحماية من عمليات حقن قاعدة البيانات mysql وهي مهمة جداً للأمان من الهكر حيث أنها ستتحقق مما يدخلة المستخدم في حقول الأدخال النماذج في الـ html .

مثال/

```
mysql_real_escape_string( $_POST[‘ user ’] ) ;
```

الفصل الثاني عشر
(دوال التعامل مع الصور)

يمكن من خلال لغة php التعامل مع الصور وذلك باستخدام مكتبة GD والتي تكون متضمنة مع الـ php بشكل افتراضي وهذه المكتبة توفر مجموعة من الدوال ومن هذه الدوال :-

- **الدالة getimagesiz**

تقوم هذه الدالة بإعطاء معلومات عن الصورة على شكل مصفوفة .

- **الدوال (imagecreatefromjpeg , imagecreatefrompng , imagecreatefromgif)**

تعمل هذه الدوال الثلاث بنفس الآلية تقريبا حيث تقوم بإنشاء مقبض للصورة عن طريق تحميل (load) الصورة من القرص الصلب , والصيغة العامة لاستدعاء هذه الدوال هي :

```
$image = imagecreatefrompng('image.png');
$image = imagecreatefromjpeg('image.jpg');
$image = imagecreatefromgif('image.gif');
```

حيث تقبل هذه الدوال الثلث وسيطا وحيدا هو مسار الصورة .

- **الدالة imagecreate**

تستخدم هذه الدالة لإنشاء صورة جديدة او مساحة عمل صورة جديدة تحتاج هذه الدالة إلى متغيرين يعبران عن عرض وارتفاع الصورة .

ملاحظة/ عند إنشاء صورة تعيد لنا هذه الدالة مقبض وهذا المقبض مهمجداً وسوف نستخدمه في الدوال القادمة بشكل مستمر .

مثال/

```
$x=@ imagecreate( 50 ,100 ) ;
```

- الدالة **imagecreatetruecolor**

تقوم بإنشاء مقبض لصورة جديدة بالبعادالمُمررة إليها كوسائط , والصيغة العامة لها هي

```
$x = imagecreatetruecolor($width, $height);
```

حيث الوسيط الأول هو عرض الصورة مقدرا بالبيكسل والثاني هو ارتفاعها .

- الدالة **imagecolorallocate**

تستخدم هذه الدالة لتعريف لون للصور لاستخدامه في الرسم وتحتاج إلى مقبض الصورة في هذه الدالة .

مثال/ نفرض أن مقبض الصورة هو $x

```
$im=imagecoloreallocate( $x , 0,0,0 ) ;
```

وكما تلاحظ في هذا المثال وضعنا درجة الإشباع للألوان (درجة إشباع أي لون هي تتراوح مابين 0 و 255) حيث أن الرقم الأول للون الأحمر والرقم الثاني للون الأخضر والرقم الثالث للون الأزرق

- الدالة **imagearc**

تستخدم هذه الدالة لرسم قطع ناقص في مساحة العمل وتحتاج إلى مقبض الصورة لكي نرسم عليها ولون الرسم .

مثال/ نفرض أن مقبض الصورة هو $x ومقبض لون الرسم هو $im

```
imagearc( $x , 100,50,100,0,360 , $im ) ;
```

- الدالة **imagechar**

تستخدم هذه الدالة لرسم خط في مساحة العملوأيضاً تحتاج إلى مقبض الصورة لكي نرسم عليها ومقبض لون الرسم .

مثال/ نفرض أن مقبض الصورة هو $x ومقبض لون الرسم هو $im

```
imagechar( $x , 1,0,0, "c" , $im ) ;
```

- **الدالة imagefontwidth**

تستخدم هذه الدالة لتحديد عرض الخطوط في مساحة العمل .

مثال/

```
$width=imagefontwidth( 20 ) ;
```

- **الدالة imagefontheight**

تستخدم هذه الدالة في تحديد ارتفاع الخطوط في مساحة العمل .

مثال/

```
$height=imagefontheight( 5 ) ;
```

- **الدوال (imagegif و imagepng و imagejpeg)**

هذه الدوال كلها لها نفس الوظيفة حيث أنها تستخدم لإخراج وحفظ الصور من مساحة العمل إلى المتصفح او ملف خارجي والفرق الوحيد بين هذه الدوال هو أنه كل دالة تحفظ الصورة بصيغة مختلفة (jpg. او gif. او png.) وتحتاج هذه الدوال إلى ثلاث وسائط الأول هو مقبض الصورة وهو إجباري والثاني هو اسم الصور الذي نريد حفظها به وهو اختياري (في حال عدم وضعه سيتم طباعة الصورة مباشرتاً إلى المتصفح) والوسيط الثالث وهو يمثل جودة الصورة وتكون بشكل نسبة مئويةوهو أيضاً اختياري (أي يمكن عدم وضعه) , والصيغة العامة لهذه الدوال هي

```
imagepng($image, $filename, $quality);
imagegif($image, $filename, $quality);
imagejpeg($image, $filename, $quality);
```

ملاحظة/ إذا أردنا أن نحدد جودة الصورة وفي نفس الوقت لا نرغب بإعطاء اسم للصورة (أي أن يتم عرض الصورةمباشرتاً في المتصفح بدون حفظها) ففي هذه الحالة يجب أن نضع بدلا من اسم الصورة القيمة null وبعده نستطيع أن نحدد جودة الصورة .

مثال/

```
<?php
$image = imagecreatetruecolor(200, 200);
#you canuse image/jpeg and image/gif for jpg and gif images
header('Content-Type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

- **الدالة imageloadfont**

تستعمل هذه الدالة لتحميل خط معين .

مثال/

```
$mf=imageloadfont( "myfont" ) ;
```

- **الدالة imagestring**

تستخدم هذه الدالة لطباعة نص على مساحة العمل والصيغ العامة لها هي :

```
imagestring($image, $font, $x, $y, $string, $color);
```

حيث أن $image هو مقبض الصورة و $font هو حجم الخط ويأخذ قيمة عددية تتراوح ما بين 1 - 5 , و $x, $y تمثل إحداثيات موقع النص و$string هو النص الذي سيطبع على مساحة العمل و$color هو مقبض اللون .

مثال/ نفرض أن مقبض الصورة هو $x ومقبض لون الرسم هو $im

```
imagestring( $x , 4 , 10 , 10 , "Ahmed" , $im ) ;
```

- الدالة **imagettftext**

هذه الدالة مشابهة للدالة السابقة , تقوم هذه الدالة بطباعة نص في مساحة العمل باستخدام خطوط من النوع ttf بأي مقاس خط وبأي زاوية , والصيغة العامة لها هي :

```
imagettftext($image, $size, $angle, $x, $y, $color, $fontfile, $text);
```

حيث أن $image هو مقبض الصورة و$size يمثل حجم الخط ويمكن أن يأخذ أي رقم كان و$angle تمثل الزاوية والزاوية يمكن أن تكون موجبة او سالبة (القيمة الموجبة تؤدي إلى الدوران عكس عقارب الساعة و القيمة السالبة تؤدي إلى الدوران مع عقارب الساعة) و$x, $y تمثل إحداثيات الصورة في موقع العمل و يمثل مقبض اللون و$fontfile تمثل مسار الخط (لأننا سنشمل خط من خارج مكتبة GD) و$text يمثل النص الذي سيطبع في مساحة العمل .

مثال/ نفرض أن مقبض الصورة هو $x ومقبض اللون هو $c

```
imagettftext( $x, 25, 0, 25, 110, $c , " /font.ttf " , "welcome" ) ;
```

- الدالة **imagecreatefromstring**

تستخدم هذه الدالة لنشاء مقبض لصورة جاهزة دون الحاجة إلى وجود ملف لها حيث يمكن ان تكون بيانات الصورة مخزنة ضامن قاعدة بيانات او باستخدام دالة base64_decode

مثال/

```
imagecreatefromstring( base64_decode(" الكود الذي يعبر عن الصورة ") ;
```

- الدالة **imagecolorat**

تستخدم هذه الدالة لإرجاع لون بكسل محدد بإحداثياته من صورة مُحددة بمقبضها , والصيغة العامة لها هي

```
$color = imagecolorat($image, $x, $y);
```

- الدالة **imagecolorsforindex**

تعيد هذه الدالة مصفوفة تحوي قيم الألوان حيث يمثل كل لونعنصراً من عناصر تلك المصفوفة .

- **الدالة imagedestroy**

تستخدم هذه الدال لهدم مقبض الصورة وتحرير الذاكرة المخزنة عليها الصورة وكل ما تحتاجه هو مقبض الصورة , والصيغة العامة لها هي :

```
imagedestroy($image);
```

- **الدالتين (imagesy و imagesx)**

تستخدم هاتان الدالتان لإعادة الطول والعرض للصورة وكل ما تحتاجانه هو مقبض الصورة وتكون الصيغة العامة لها بالشكل التالي (على فرض أن $x هو مقبض الصورة) :

```
imagesx($x);
```

```
imagesy($x);
```

- **الدالة imagefill**

تقوم هذه الدالة بتلوين منطقة محددة بلون واحد أي كما تقوم أداة التعبئة في برامج الرسم , حيث أنها تقبل أربع وسائط الوسيط الأول هو مقبض الصورة والثاني والثالث يمثلان إحداثيات المنطقة التي سيتم تلوينها والوسيط الرابع هو يمثل مقبض اللون الذي تعيده الدالة imagecolorallocate , والصيغة العامة لها هي :

```
imagefill($image, $x, $y, $color);
```

- **الدالة imagefilledrectangle**

تقوم هذه الدالة بملء مستطيل بلون محدد , والصيغة العامة لها هي :

```
imagefilledrectangle($image, $x1, $y1, $x2, $y2, $color);
```

حيث أن $image تمثل مقبض الصورة $x1, $y1 هي إحداثيات الزاوية اليسرى العليا و $x2, $y2هي إحداثيات الزاوية اليمنى السفلى و$color يمثل مقبض اللون الذي تعيده الدالة imagecolorallocate

- الدالة **imagerotate**

تستخدم هذه الدالة لتدوير الصورة حول مركزها , والصيغة العامة لها هي :

```
imagerotate($image, $angle, $bg_color);
```

حيث الزاوية بالدرجات و$bg_color هو اللون الذي سيتم وضعه مكان الفراغ نتيجة التدوير .

مثال/ نفرض أن مقبض الصورة هو $x

```
imagerotate($x, 45, 0xffffff);
```

- الدالة **imagesetpixel**

تقوم هذه الدالة بتحديد لون بكسل معين بإحداثيات $x, $y , والصيغة العامة لها هي :

```
imagesetpixel($image, $x, $y, $color);
```

حيث أن $image هو مقبض الصورة و$color هو مقبض اللون .

- الدالة **imageline**

تستخدم هذه الدالة لرسم مستقيمات بين نقطتين محددتين , والصيغة العامة لها هي :

```
imageline($image, $x1, $y1, $x2, $y2, $color);
```

حيث أن $image هو مقبض الصورة و $x1, $y1 هي إحداثيات نقطة البداية و $x2, $y2 هي إحداثيات نقطة النهاية و $color هو مقبض اللون .

- الدالة **imagesetthickness**

تستخدم هذه الدالة لتحديد سمك خط الرسم وهي تأخذ وسيطين الأول يمثل مقبض الصورة والثاني يمثل سمك الخطمقدراً بالبيكسل

مثال/ على فرض أن مقبض الصورة هو $x

```
imagesetthickness($x, 5);
```

- **الدالة imagettfbbox**

تقوم هذه الدالة بإعادة مصفوفة تحوي إحداثيات نص باستخدام خط معين والصيغة العامة لها هي

```
imagettfbbox($size, $angle, $fontfile, $text);
```

تفيد هذه الدالة بحساب أبعاد أي نص مكتوب بأي خط لاستخدامها في محاذاة النص (توسيطمثلاً)

- **الدالة imagecopy**

تستخدم هذه الدالة لنصخ صورة إلى صورة أخرى وتقبل الوسائط التالية :

1- الوسيط $dst_im هو الصورة التي سيتم النسخ إليها (الصورة الهدف) .
2- الوسيط $src_im الصورة التي سيتم النسخ منها .
3- الوسائط $dst_x, $dst_y, $src_x, $src_y هي إحداثيات بداية انسخ واللصق .
4- الوسيطين $src_w, $src_h هم عرض وطول الجزء المنسوخ .

وستكون الصيغة العامة لهذه الدالة بهذا الشكل :

```
imagecopy($dst_im, $src_im, $dst_x, $dst_y, $src_x, $src_y,$src_w, $src_h);
```

- **الدالة imagecopyresized**

تقوم هذه الدالة بنسخ جزء من الصورة ولصقه على صورة أخرى مع تغيير أبعاده والصيغة العامة لها هي :

```
imagecopyresized($dst_image, $src_image, $dst_x, $dst_y, $src_x,$src_y, $dst_w,
$dst_h, $src_w, $src_h);
```

وهذه الوسائط هي نفسها الموضحة بالدالة السابقة

- **الدالة imagefilter**

تستخدم هذه الدالة لتطبيق تأثيرات على الصور (فلاتر) , وتأخذ هذه الدالة عدد متغير من الوسائط بحسب التأثير المُمرر إليها ولكن كما هومُعتاد يكون الوسيط الأول هو مقبض الصورة والثاني هو الثابت الخاص بالفلتر , المُستخدم و باقي الوسائط هي وسائط تختلف حسب التأثير المُستخدم , والصيغة العامة للدالة هي :

```
imagefilter($image, $filtertype, $arg1, $arg2, $arg3);
```

ومن تأثيرات الفلاتر المستخدمة مع هذه الدالة هي :-

1- تأثير الإضاءة

عند تمرير الثابت IMG_FILTER_BRIGHTNESS إلى الدالة السابقة يمكن تغيير الإضاءة في الصور وعند استخدام هذا التأثير يجب تمرير وسيط آخر هو قيمة الإضاءة التي تتراوح قيمتها بين 255- إلى 255 حيث أن القيمة 255 تمثل إضاءة كاملة (اللون الأبيض) والقيمة 255- تمثل اللون الأسود والقيمة 0 تبقي الإضاءة على حالها .

مثال/

```
<?php
$image = imagecreatefromjpeg('image.jpg');
imagefilter($image, IMG_FILTER_BRIGHTNESS, 100);
header('Content-Type: image/png');
imagejpeg($image);
?>
```

2- تطبيق تأثير الضبابية blur

وذلك عند استخدام الثابت IMG_FILTER_SELECTIVE_BLUR أو الثابت IMG_FILTER_GAUSSIAN_BLUR , ولا داعي لاستخدام أي وسيط إضافي .

والجدول التالي يوضح الفلاتر المدعومة من قبل الدالة imagefilter :-

الثابت	الشرح
IMG_FILTER_NEGATE	عكس جميع ألوان الصورة
IMG_FILTER_GRAYSCALE	تحويل الصورة إلى صورة رمادية (أبيض وأسود)
IMG_FILTER_BRIGHTNESS	كما في الفقرة السابقة تستخدم للتحكم بالإضاءة , وتقبل وسيطاً أضافياً هو مقدار الإضاءة
IMG_FILTER_CONTRAST	تغيير تباين الصورة , تقبل وسيطاً ضافياً هو مقدار التباين
IMG_FILTER_EDGEDETECT	تطبيق خوارزمية لإظهار حواف مكونات الصورة
IMG_FILTER_GAUSSIAN_BLUR	إضافة تأثير الضبابية للصورة باستخدام خوارزمية blur gaussian
IMG_FILTER_SELECTIVE_BLUR	إضافة تأثير الضبابية للصورة
IMG_FILTER_PIXELATE	إضافة تأثير pixelate للصورة

الفصل الثالث عشر

(دوال التعامل مع سيرفر FTP)

- الدالة **ftp_connect**

تقوم هذه الدالة بإجراء اتصال مع سيرفر FTP فإذا تم الاتصال بنجاح تعود بمقبض لهذا الاتصال لنتمكن من استخدامه في بقية العمليات .

مثال/

```
$ftp=ftp_connect( "www.example.com" ) ;
```

- الدالة **ftp_login**

تقوم هذه الدالة بتسجيل دخول المستخدم إلى السيرفر وتحتاج هذه الدالة إلى مقبض الاتصال والاسم وكلمة السر وتعود بالقيمة true إذا تم الاتصال بنجاح .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$log=ftp_login( $ftp , "user" , "password" ) ;
```

- الدالة **ftp_pwd**

تعود هذه الدالة باسم المجلد الحالي الذي نتعامل معه وتحتاج هذه الدالة فقط إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$dir=ftp_pwd( $ftp ) ;
```

- الدالة **ftp_chdir**

تقوم هذه الدالة بتغيير المجلد الحالي في السيرفر وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$chdir=ftp_chdir( $ftp , "www" ) ;
```

في هذا المثال نطلب الاتجاه إلى المجلد www

- الدالة **ftp_mkdir**

تقوم هذه الدالة بإنشاء مجلد جديد في المسار المحدد وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$mk=ftp_mkdir( $ftp , "sss" ) ;
```

- الدالة **ftp_rmdir**

تقوم هذه الدالة بحذف مجلد معين في المسار المحدد وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$rm=ftp_rmdir( $ftp , "sss" ) ;
```

- الدالة **ftp_nlist**

تقوم هذه الدالة بالعودة بأسماء الملفات والمجلدات في المسار المحدد على شكل مصفوفة وتحتاج إلى مقبض عملية الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$list=ftp_nlist( $ftp , "." ) ;
```

- **الدالة ftp_get**

تستخدم هذه الدالة لتنزيل ملف من سيرفر FTP إلى جهازك او موقعك وتحتاج إلى مقبض الاتصال وأيضاً تحتاج إلى نوع النقل هل هو FTP_ASCII او FTP_BINARY

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
ftp_get( $ftp , "/tmp/data.bin" , "/pub/data.bin" , FTP_BINARY ) ;
```

- **الدالة ftp_put**

تقوم هذه الدالة بنقل الملف من جهازك إلى سيرفر ftp وتحتاج إلى مقبض الاتصال وأيضاً تحتاج إلى نوع النقل هل هو FTP_ASCII او FTP_BINARY

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
ftp_get( $ftp , "/tmp/data.bin" , "/pub/data.bin" , FTP_BINARY ) ;
```

- **الدالة ftp_size**

تعيد هذه الدالة حجم الملف المحدد وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$size=ftp_size( $ftp , "file.text" ) ;
```

- **الدالة ftp_rename**

تقوم هذه الدالة بتغيير الاسم للملف المحدد وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$x=ftp_rename( $ftp , "file.txt" , "fileto.txt" ) ;
```

- **الدالة ftp_delete**

تقوم هذه الدالة بحذف الملف المحدد وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$d=ftp_delete( $ftp , "file.txt" ) ;
```

- **الدالة ftp_site**

تقوم هذه الدالة بإرسال الأوامر إلى السيرفر وتحتاج إلى مقبض الاتصال

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$x=ftp_site( $ftp , "cd" ) ;
```

- **الدالة ftp_quit**

تقوم هذه الدالة بإغلاق الاتصال مع السيرفر وتحتاج إلى مقبض الاتصال .

مثال/ نفرض أن مقبض الاتصال هو $ftp

```
$close=ftp_quit( $ftp ) ;
```

الفصل الرابع عشر
(اشتمال الملفات والتعامل معها)

أولاً :- اشتمال الملفات (include files)

قد يكون لديك في برنامجك متغير متكرر في أكثر من صفحة أو رسالة خطأ معينة أو تريد إدراج نص كبير الحجم في صفحات متعددة , هنا يمكن اشتمال ملفات داخل ملفات php هذه الملفات قد تحتوي على نصوص أو اكواد html أو كود php , والصيغة العامة هي :-

```
include ( file name ) ;
```

مثال :-

قم بفتح ملف نصي وأكتب فيه ما تشاء ثم أحفظه باسم a.txt ثم قم بإنشاء ملف php واكتب فيه هذا الكود

```
<?php
include ( "a.txt" ) ;
?>
```

انقلها إلى مجلد السيرفر ... شغل ملف php وأنظر النتيجة .

يمكنك أن تنشأ ملف php وتحتفظ فيه في تعريف دالة (**function**) وعند الحاجة تستدعيه في أي مكان .

لاحظ هنا أضفنا فقط اسم الملف النصي باعتبار إنه موجود في نفس المجلد ولكن إن كان في مجلد آخر يجب إضافة مسار الملف كاملاً .

ملاحظه/ الدالة **require** هي تماماً تعمل نفس الدالة **include** لكن الفرق بينهم هو انه اذا جلبنا صفحة وكان مسارها خطأ او الصفحة غير موجودة فأنه سيظهر خطأ حيث انه في الدالة **include** سوف يظهر الخطأ ويستمر في تنفيذ باقي كود الصفحة أما في الدالة **require** فأنه سيوقف تنفيذ باقي الكود الموجود في الصفحة , وهناك دوال أخرى تستخدم لضمين الملفات مثل

الدالة include_once وهي تستخدم لتضمن الملف مرة واحدة أي إذا كنا قد طلبنا الملف مرتين فأنه سيجلبه مرة واحدة فقط ونفس الشيء بالنسبة للدالة require_once .

دالة عرض مصدر الكود

يمكن عرض (طباعة) كود صفحة معين وذلك باستخدم الدالة (**show_source**) وسيتم عرض الكود بشكل منسق ومرتب (كما هو مكتوب) .

مثال :-

```
<?php
show_source ( "style.css" ) ;
?>
```

ثانياً :- التعامل مع الملفات و المجلدات

أولاً :- التعامل مع الملفات

- **الدالة basename**

تقوم هذه الدالة باستخلاص اسم الملف من مسار معين

مثال/

```
echo basename( "www.example.com/help/index.php" ) ;
```

- **الدالة chmod**

تستعمل هذه الدالة للتعديل على التراخيص للملفات او المجلدات سواء أن نتيحها للقراءة او للكتابة او القراءة والكتابة

ملاحظة/ عند استخدام برنامج ftp في إرسال الملفات إلى الانترنت فأن أهم ترخيصين هما 755 و777 , حيث أن الترخيص 755 يعطى لملفات السكربتات والترخيص 777 يعطى للملفات والمجلدات التي يراد أتاحتها للكتابة .

مثال/ لو أردنا تغيير الترخيص للمجلد المسمى (test) سوف نكتب الترخيصمسبوقاً بصفر هكذا :-

```
chmod( "test" , 0755 ) ;
```

ملاحظة/ تكون الصلاحية (الترخيص) مكونة من أربعة أرقام الرقم الأول هو صفر , أما الأرقام الثلاث الباقية هي عبارة عن الصلاحيات (تراخيص) للمستخدم ولمجوعة المستخدم و لبقية المستخدمين على التوالي و بالترتيب , الجدول التالي يوضح الأرقام والصلاحيات المقابلة لها

الرقم	الصلاحية
الرقم 0	يشير إلى عدم إعطاء أي صلاحية .
الرقم 1	يشير إلى إعطاء صلاحية التنفيذ فقط .
الرقم 2	يشير إلى إعطاء صلاحية الكتابة فقط .
الرقم 3	يشير إلى إعطاء صلاحية الكتابة والتنفيذ .
الرقم 4	يشير إلى إعطاء صلاحية القراءة فقط .
الرقم 5	يشير إلى إعطاء صلاحية القراءة والتنفيذ .
الرقم 6	يشير إلى إعطاء صلاحية القراءة والكتابة .
الرقم 7	يشير إلى إعطاء صلاحية القراءة والكتابة والتنفيذ .

ومن الجدول السابق يمكن أن نكون أرقام الصلاحيات التي نحتاجهلمثلاً الرقم (0600) يشير إلى إمكانية القراءة والكتابة للمستخدم ولا شيء لبقية المستخدمين , والرقم (0755) يشير إلى إمكانية القراءة والكتابة والتنفيذ للمال والقراءة والتنفيذ لبقية المستخدمين .

- **الدالة chown**

تستخدم لإعطاء ملكية او لتغيير ملكية ملف او مجلد .

ملاحظة / هذه الدالة لا تستخدم إلا من قبل المدير للملفات .

مثال/

```
chown ( الملف او المجلد , معرف المستخدم ) ;
```

- **الدالة copy**

تستخدم لإنشاء نسخة احتياطية لملف يحتوي على معلومات مهمة للحفاظ عليها

مثال/

```
copy( "arab1.html" , "arab1.php" ) ;
```

حيث هنا ننسخ الملف arab1.html إلى الملف arab1.php في نفس المجلد .

- **الدالة unlink**

تستخدم هذه الدالة لحذف ملف او مجلد ونحتاج فقط إلى مسار هذا الملف او المجلد (وسنتناول موضوع المجلدات بعد قليل) .

مثال/

```
unlink( " file.txt" ) ;
```

ملاحظة/ هنا في هذا المثال كان الملف في نفس مجلد البرنامج وإن لم يكن في نفس مجلد البرنامج فيجب كتابة المساكاملاً .

- **الدالة diskfreespace**

تعطي هذه الدالة المساحة المتبقية لدينا في القرص الصلب بالبايت .

مثال/

```
echo diskfreespace( "/" ) ;
```

ويمكن أن نكتب اسم المجلد الفرعي إن كان لدينا لنحصل على مساحته .

- الدالة **memory_get_usage**

تفيد هذه الدالة في معرفة حجم الملف او السكربت الذي نعمل عليه وتعود لنا برقم يمثل حجم السكربت .

مثال /

```
echo memory_get_usage( ) ;
```

لاحظ بأن هذه الدالة تحسب ما فوقها فقط ولكي نحسب حجم كامل الملف (السكربت) يجب ان نكتبها في اسفل الكود .

- الدالة **filesize**

تستخدم هذه الدالة لمعرفة حجم الملف بالبايت وتحتاج فقط إلى اسم الملف .

مثال/

```
echo filesize( "pic.php" ) ;
```

- الدالة **filetype**

هذه الدالة تعيد لنا نوع الملف وتحتاج فقط اسم الملف .

مثال/

```
echo filetype( "pic.gif" ) ;
```

- الدالة **file**

تستعمل لقراءة الملفات واستخلاص البيانات منها , حيث ستضع الملفكاملاًوسطراًسطراً داخل مصفوفة بحيث يصبح كل سطر في متغير من متغيرات المصفوفة .

مثال/

```
$arr = file( “file.txt”) ;

echo arr[0] ;
```

في هذا المثال سيطبع السطر الاول من الملف

- **الدالة file_exists**

لكي نتعامل مع ملف سواء بالقراءة او الكتابة او غيرها لابد من معرفة هل الملف موجود في الموقع السليم أم لا , لأننا إذا حاولنا أن نتعامل معه ولم يكنموجوداً فسوف نقع في أخطاء , وهذه الدالة نعطيها اسم الملف فتعيد لنا القيمة 1 إذا كان الملف موجود او أي قيمة أخرى إذا لم يكن موجود .

مثال/

```
if( file_exists( "pic.jpg") ) { echo "الملف موجود"
} else{ echo "الملف غير موجود" }
```

- **الدالة file_get_contents**

تستخدم هذه الدالة لقراءة ملف بأكمله على شكل سلسلة نصية و يُمرر لها مسار الملفكاملاً .

مثال/

```
file_get_contents('login.json')
```

- **الدالة file_put_contents**

تقوم هذه الدالة بكتابة بيانات التي تمرر لها إلى ملف ما وهي تأخذ وسيطين الأول يمثل مسار الملف والثاني يمثل البيانات التي ستضاف للملف , وتقوم هذه الدالة بإنشاء الملف إذا لم يكن موجودا , وفي حال وجوده تقوم بمسح جميع محتوياته .

- **الدالة fopen**

لكي نتعامل مع أي ملف لابد من أن نقوم بفتحه وهذه الدالة تفتح الملف وتحتاج هذه الدالة إلى ثلاثة متغيرات حيث أن المتغير الثالث اختياري ويمكن عدم كتابته وتعود لنا هذه الدالة برقم يدعى مقبض الملف عن طريقه نتعامل مع الدوال الأخرى وإليك المتغيرات :-

المتغير الأول :- اسم الملف المراد فتحه .

المتغير الثاني :- ما هي الحالة التي تريد أن تفتح الملف عليها (للقراءة فقط او للقراءة والكتابة او للكتابة فقط) .

المتغير الثالث :- وهو اختياري ويحدد هل تريد استخدام المجلدات المحددةمسبقاً في خيارات php والقيمة 1 تعني نعم .

* المتغيرات التي تعبر عن حالات فتح الملف هي

المتغير	العمل
r	فتح الملف للقراءة فقط ووضع المؤشر في البداية .
r+	فتح الملف للقراءة والكتابة ووضع المؤشر في البداية .
w	فتح الملف للكتابة فقط ووضع المؤشر في البداية ومسح جميع البيانات , وإذا لم يكن الملف موجود ستقوم الدالة بإنشائه .
w+	فتح الملف للكتابة والقراءة ووضع المؤشر في البداية ومسح جميع البيانات , وإذا لم يكن الملف موجود ستقوم الدالة بإنشائه .
a	فتح الملف للكتابة فقط ووضع المؤشر في النهاية , وإذا لم يكن الملف موجود ستقوم الدالة بإنشائه .
a+	فتح الملف للقراءة والكتابة ووضع المؤشر في النهاية , وإذا لم يكن الملف موجود ستقوم الدالة بإنشائه .

مثال/

```
$fp=fopen( "file.txt" , "r" ) ;
```

مثال/

```
$fp=fopen( "http://www.example.com/file.txt" , "r" ) ;
```

- **الدالة fclose**

وهي تستخدم لإغلاق الملف المفتوح حيث إن أبقيت الملف مفتوح فسوف يستهلك جزء من الذاكرة وكل ما نحتاجه هنا هو مقبض الملف المفتوح الذي تأتينا به الدالة fopen

مثال/

```
$fp=fopen( "file.txt" , "r" ) ;
fclose( $fp ) ;
```

- **الدالة fread**

تستخدم هذه الدالة لقراءة ملف معين بعد فتحه وهي تقرأ جزء معين من الملف يحدده المبرمج مما يقلل من الذاكرةفمثلاً لو كان لدينا ملف طوله 100 بايت وفتحناه ومن ثم قرأناه بهذه الدالة وحددنا له الحجم 20 بايت فعند استدعائها سوف تحضر لنا أول 20 بايت وعند استدعائها مرة أخرى لنفس الملف تحضر الـ 20 بايت التالية وهكذا حتى نصل إلى نهاية الملف , حيث أن الدالة تحتاج إلى متغيرين الأول مقبض الملف والثاني الحجم الذي نريد قراءته .

مثال/

```
$fp = fopen( "file.txt" , "r" ) ;
$contents = fread( $fp , 20 ) ;
fclose( $fp ) ;
echo $content;
```

لاحظ بأن هذا المثال سيطبع أول 20 حرف من الملف

مثال / لقراءة كامل الملف بحسب حجمه

```
$s = filesize("file.html") ;
$fp=fopen( "file.txt" , "r" ) ;
$contents=fread( $fp , $s ) ;
fclose( $fp ) ;
```

لاحظ ان الملف اذا كان فارغ فأن الدالة filesize ستعيد الرقم 0 وهنا سيحدث خطأ عند استخدامها داخل الدالة fread لذلك يجب أن نتأكد ان الملف غير فارغ .

- **الدالة fgets**

هذه الدالة هيتماماً مثل الدالة السابقة (fread) والفرق الوحيد هو أن هذه الدالة لا نحدد لها حجم بالبايت لتقرأه لأنها ستأخذ بشكل افتراضي الحجم 1024 بايت , وكلا الدالتين تتوقفان بعد قراءة الملف بالكامل

- **الدالة feof**

وهي تفيدنا هل تم قراءة الملف بالكامل أي هل وصلنا إلى نهاية الملف ونحتاج فقط إلى مقبض الملف وسوف يعيد لنا القيمة true إذا كنا قد وصلنا إلى نهاية الملف .

مثال/

```
feof( $fp ) ;
```

- **الدالة fwrite**

هذه الدالة تقوم بعملية الكتابة و هي تحتاج إلى المتغيرات التالية :-

1- مقبض الملف المراد الكتابة عليه .
2- النص المارد كتابته .
3- وهذا المتغير اختياري وهو حجم النص المراد كتابتهو غالباً لا يستخدم .

وسيتم الكتابة في داخل الملف في المكان الذي توقف فيه مؤشر قراءة الملف .

مثال/

```
fwrite( $fp , "My Program" ) ;
```

ملاحظة/ ستقوم هذه الدالة بطباعة رسالة خطأ عند عدم توفر صلاحيات للكتابة على الملف .

- **الدالة fputs**

هذه الدالة هيتماماً نفس الدالة السابقة (fwrite) وتأختتماماً نفس وسائطها .

- **الدالة fseek**

تستخدم هذه الدالة لتغيير مكان المؤشر سواء عند القراءة أو الكتابة وهي تقبل وسيطين إجباريين , الأول هو مقبض الملف والثاني هو المكان الذي سوف يتم وضاع المؤشر عنده , وتكون الصيغة العامة لها بهذا الشكل

```
fseek($handle, $offset);
```

- **الدالة rename**

وهي تستخدم لتغيير اسم ملف وتحتاج إلى متغيرين الأول الاسم القديم والثاني الاسم الجديد .

مثال/

```
echo rename( "backup.gif" , "backup1.gif") ;
```

- الدالة **realpath**

تستعمل هذه الدالة للتعرف على المسار الكامل والحقيقي للملف المحدد .

مثال/

```
echo realpath( "backup.gif" ) ;
```

- الدالة **tempnam**

تقوم هذه الدالة بإنشاء ملف مؤقت وحيد أي اسمه غير موجودمسبقاً .

مثال/

```
$x=tempnam( "  " , "sag" ) ;
echo $x ;
echo "</p>" ;
$x=tempnam( "  " , "sag" ) ;
echo $x ;
```

حيث أننا في هذا المثال استدعينا هذه الدالة مرتين أول ثلاث أحرف منها sag وستعطي لكل واحد منها رقم مميز لكي لا يكونا نفس الاسم .

- الدالة **filectime**

تستخدم هذه الدالة للحصول على آخر وقت لتغيير ملف ما وتقبل وسيط واحد بين قوسيها وهو مسار الملف , وتعيد هذه الدالة جميع التغيرات على ملف سواء على محتوياته أم على صلاحيات الوصول إليه أم تغيير المستخدم المالك له .

مثال/

```
filectime('file1.txt');
```

- الدالة filemtime

تستخدم هذه الدالة للحصول على آخر وقت لآخر تعديل على الملف وتقبل هذه الدالة بين قوسيها وسيط واحد وهو مسار الملف , ولاحظ أن هذه الدالة تشير إلى آخر تعديل في محتويات الملف فقط

مثال/

```
echo date("m/d/Y H:i:s", filemtime('file1.txt'));
```

- الدالة fileatime

تعيد هذه الدالة بصمة وقت آخر وصول للملف او false في حال فشلها وتقبل هذه الدالة بين قوسيها وسيط واحد وهو مسار الملف .

مثال/

```
fileatime('file1.txt');
```

- الدالة is_readable

تستخدم هذه الدالة لمعرفة إمكانية القراءة على ملف او قيد وتعيد القيمة true في حال نجاحها والقيمة false في حال فشلها , وتقبل بين قوسيها وسيط واحد وهو مسار الملف .

مثال/

```
is_readable('file1.txt');
```

- الدالة is_writeable

تستخدم هذه الدالة لمعرفة إمكانية الكتابة على ملف او قيد وتعيد القيمة true في حال نجاحها والقيمة false في حال فشلها , وتقبل بين قوسيها وسيط واحد وهو مسار الملف .

مثال/

```
is_writable('file1.txt');
```

- الدالة **is_ executable**

تستخدم هذه الدالة لمعرفة إمكانية التنفيذ على ملف او قيد وتعيد القيمة true في حال نجاحها والقيمة false في حال فشلها , وتقبل بين قوسيها وسيط واحد وهو مسار الملف .

مثال/

```
is_executable('file1.txt');
```

ثانياً :- التعامل مع المجلدات

- الدالة **dirname**

تقوم هذه الدالة باستخلاص اسم المجلد من المسار .

مثال/

```
echo dirname( "www.example.com/help/index.php" ) ;
```

- الدالة **opendir**

تستخدم هذه الدالة للحصول على مقبض المجلد حيث تقبل هذه الدالةوسيطاً واحد هو مسار المجلد , والصيغة العامة لها هي

```
$resorce = opendir($path);
```

- الدالة **closedir**

تستخدم هذه الدالة لغلق المجلد الذي فتحناه بواسطة الدالة السابقة وهي تأخذ وسيط واحد هو مقبض الاتصال الذي تجلبه الدالة السابقة , والصيغة العامة لها هي

```
closedir($handle);
```

- الدالة **readdir**

تستخدم هذه الدالة لقراءة القيد التالي من مجلد تم إنشاء مقبضه بواسطة الدالة opendir حيث تقوم هذه الدالة بقراءة قيود الملفات على التتالي وحسب ترتيب نظام الملفات المُستخدم تقبل هذه الدالة وسيطا واحدا هو مقبض المجلد و تُعيد القيد (اسم الملف أو المجلد) , وللمرور على جميع قيود المجلد نستخدم حلقة التكرار while ولتطبيق الدوال الثلث السابقة نجرب المثال التالي

```
<?php
$dir = opendir('folder');
while (($file = readdir($dir) )!== false)
{
echo $file.'<br>';
}
closedir($dir);
?>
```

- الدالة **mkdir**

تقوم هذه الدالة بإنشاء المجلد وتحتاج إلى متغيرين الأول اسم المجلد والثاني الترخيص الذي تود إعطاءه له .

مثال/

```
echo mkdir( "backup" , 0777 ) ;
```

وهنا أعطيناه ترخيص للدخول والقراءة والكتابة والتنفيذ .

- الدالة **is_dir**

عندما نريد أن ننشأ مجلد ويكون موجود يحدث خطأ يعطل البرنامج وهذه الدالة تخبرنا أن كان المجلد موجود ولا تحتاج إلا إلى اسم المجلد وإن كان موجود تعود لنا بالقيمة 1 وإن لم يكن موجود تكون القيمة 0 .

مثال/

```
echo is_dir( "mydir" ) ;
```

- الدالة rmdir

وهي تستخدم لحذف المجلد وكل ما تحتاجه هو اسم المجلد , وقبل استخدامها من المفضل استخدام الدالة is_dir للتأكد من أن الملف موجود .

مثال/

```
echo rmdir( "backup" ) ;
```

ملاحظة/ يجب أن يكون المجلد المراد حذفه فارغا أما إذا كان المجلد يحتوي أي ملف او مجلد فرعي , فلن يتم تنفيذ هذه التعليمة و سيتم توليد رسالة خطأ لكن إذا أردنا حذف مجلد يحوي ملفات ومجلدات فرعية , فيجب علينا أولاً أن نقوم بحذف جميع محتوياته قبل محاولة استدعاء هذه الدالة ,وأيضاً يجب الانتباه الى ان التصريح المعطى للملف يسمح بحذفه .

ثالثاً :- التعامل مع الملفات المضغوطة zip

في البداية للتعامل مع هذا النوع من الملفات يجب ان نفعلها من خلال الملف php.ini الذي يكون مرفق مع السرفر حيث سنبحث عن العبارة (extension=php_zip.dll) في داخل الملف سنجد قبل هذه العبارة (في بداية السطر) هذه العلامة (;) الفارزة المنقوطة وهذا يدل على انها لا تعمل وكل ما علينا فعله هو ان نحذف هذه الفارزة الموجودة في بداية السطر لتفعيلها .

- zip_open

تستخدم هذه الدالة للفتح الملف المضغوط وكل ما علينا فعله هو ان نمرر لها مسار هذا الملف بين قوسيها , وستعيد لنا هذه الدالة مقبض لعملية الفتح .

مثال/

```
$o = zip_open("file.zip");
```

- zip_close

تستخدم هذه الدالة لتحرير مقبض الفتح الذي تعيده الدالة zip_open وهي لا تحتاج إلا الى اسم المقبض بين قوسيها .

مثال/

```
$o = zip_open("file.zip");

zip_close( $o );
```

- **zip_read**

تستخدم هذه الدالة لقراءة الملفات الموجودة بداخل الملف المضغوط ونضع بين قوسيها المقبض الذي عادت به إلينا الدالة zip_read , ولاحظ بأنه اذا اردنا الوصول الى ملف محدد موجود في داخل الملف المضغود فيجب ان نستخدم حلقة تكرار لتدور على كل الملفات الموجودة ومن ثم نستخدم الدالة المناسبة للتعامل مع هذا الملف وسنشرح هذه الدوال بعد قليل .

مثال/

```
$o = zip_open("file.zip");
if( $o ){
$e = zip_read( $o );
while( $e ){
// هنا نكتب الدوال التي نحتاجها للتعامل مع الملفات الداخليه وهذه الدوال سنشرحها بعد قليل
}
}
```

لاحظ بأننا تحققنا من مقبض الأتصال $o لأنه اذا كان المسار الممرر الى الدالة zip_open خطأ ستعيد لنا false وستسبب لنا مشكله لذلك وضعنا هذا الشرط

- **zip_entry_name**

تستخدم هذه الدالة لقراءة اسماء الملفات الموجودة داخل الملف المضغوط .

مثال/

```
$o = zip_open("file.zip");
if( $o ){
$e = zip_read( $o );
while( $e ){
$n = zip_entry_name( $e );
echo $n . "<br>" ;
$e = zip_read( $o );
} }
```

في هذا المثال سيقراء كل اسماء الملفات الموجودة داخل الملف المضغوط .

- **zip_entry_filesize**

تستخدم هذه الدالة لقراءة حجم الملف الداخلي الموجود داخل الملف المضغوط .

مثال/ لجلب أحجام كل الملفات الموجودة في الملف المضغوط .

```
$o = zip_open("file.zip");
if( $o ){
$e = zip_read( $o );
while( $e ){
$n = zip_entry_filesize( $e );
echo $n . "<br>" ;
$e = zip_read( $o );
}
}
```

الفصل الخامس عشر
(تصيد الأخطاء)

إذا كتب المستخدم جملة في مربع نص يحتوي على عدة كلمات وتريد أن تتأكد من وجود كلمة معينة وسط هذه الجملة أو معرفة صحة كتابة البريد الاليكتروني المدخل أو ... الخ , نحتاج إلى الدالة **ereg** .

- **الدالة ereg**

تستخدم هذه الدالة لمعرفة وجود كلمة في متغير ما , والصيغة العامة لها هي :-

```
ereg ( " الكلمة " , اسم المتغير )
```

مثال :-

```
$word = "one , two , three , one " ;
if ( ereg ("one" , $word , $rog ) ) { echo $rog[0] ; }
if ( ereg ("two" , $word , $rog ) ) { echo $rog[0] ; }
```

نلاحظ في المثال أعلاه عدة أمور :-

1- استخدام الدالة **ereg** في أداة الشرط **if** واستخدامها في أكثر من مكان للبحث عن أكثر من كلمة .

2- في داخل الدالة **ereg** وضعنا بعد اسم المتغير اسم مصفوفة وذلك لتخزين الكلمة في مصفوفة , ويمكن عدم كتابة هذه المصفوفة .

3- وجود أكثر من كلمة one في المتغير word ومع ذلك فهو لا يطبع سوى كلمة واحدة وذلك لأن الدالة **ereg** عندما تجد أول كلمة سوف تعطي النتيجة true وبذلك يتحقق الشرط ولا تهتم بعدد الكلمات .

ملاحظة/ الدالة **ereg** حساسة لحالة الأحرف الكبيرة والصغيرة للكلمة المطلوب البحث عنها فأن (One) تختلف عن (one) وأيضاً يجب الانتباه للفواصل حيث لو كتبنا (" one") (وضعنا

فاصلة قبل الكلمة) ففي هذه الحالة لا تستطيع الدالة إيجاد الكلمة لو كانت من ضمن كلام مثل (onedy) ولكن يمكن إيجاد الكلمة (one) من ضمن هذا الكلام إذا حذفنا الفاصلة قبل الكلمة ("one") .

ملاحظة/ يمكن وضع مكان الكلمة المراد البحث عنها في الدالة **ereg** اسم متغير معرفـمسبقاً

مثال :-

```
$d = "one" ;
$word = "one , two , three" ;
if (ereg ($d , $word ) { echo " 'one' لقد وجدت العدد " ; }
```

ملاحظة/ تعتبر للنقطة (.) قداسة في الدالة **ereg** وعند وضعها فأنه سيفهمها على إنها مكان لحرف أو فراغ ولكي نلغي هذه القداسة وتُفهم php على إنها رمز النقطة (.) يجب إضافة العلامة (\ ) قبل النقطة , مثل :- (one\.)

صناعة فئة حروف [xyz]

نقصد بذلك إننا نحدد نطاق معين من الكلمة من الممكن أن يكون في هذا النطاق أي حروف من الفئة التي أقوم بتحديدها .

مثال :-

```
$y = "how are you?" ;

if ( ereg ( "h[oe]" , $y ) { echo "true" ; }
```

هنا فإنه سوف يبحث عن أي كلمة تبدأ بالحرف h ومن ثم يتبعها أحد هاذين الحرفين o أو e , وإن أردنا البحث عن كلمة تبدأ بالحرف h ولكن يجب أن لا يتبعها هاذين الحرفين فإننا نضيف العلامة (^) قبل الحرفين "h[^oe]" .

ملاحظة/ يمكننا استعمال اختصارات لبعض الأمورفمثلاً إذا كنا نريد كلمة لا تحتوي على أي رقم كنا سنكتب كالتالي [123456789 ^] ويمكن استعمال اختصار للموضوع كالتالي [^ 0 - 9] , وحتى إذا أردنا أن يتأكد من وجود الرقم فعلينا مسح هذه العلامة (^) , وهذه الاختصارات في الجدول في الأسفل تشمل أحرف وأشياء أخرى .

الاختصار	المطابق له	معناه و وضيفته
\d	[0 - 9]	أي رقم من 0 إلى 9
\D	[^ 0 - 9]	ممنوع الأرقام من 0 إلى 9
\w	[0-9 A-Z a-z-]	أي رقم أو أي حرف كبير أو صغير أو علامة -
\W	[^ 0-9 A-Z a-z-]	عكس السابقة
\s	[\t \n \r]	يقبل مسافة أو سطر جديد أو علامة جدول tab
\S	[^\t \n \r]	عكس السابقة

ملاحظه/ لتحديد مكان الكلمة إن كانت في البداية أو في نهاية النص حيث إن العلامة (^) لتحديد المكان لبداية الجملة والعلامة ($) لنهاية الجملة .

مثال :-

```
$x = "how are you" ;
if (ereg ("^h" , $x) {echo "true" ; }
```

مثال :-

```
$x = "how are you" ;
if (ereg ("h$" , $x) {echo "true" ; }
```

يمكنأيضاً الاختيار إذا ما كان واحد من نمطينصحيحاً بواسطة العلامة (|) .

مثال :-

```
$x = "how are you" ;
if (ereg ("^g | y$" , $x) {echo "true" ; }
```

سيعتبر في هذا المثال إن الشرط متحقق .

العلامات (* و + و ?)

باستخدام هذه العلامات يمكن تحديد إذا ما كان حرف أو جملة متكررة بعدد من المرات أو مرة واحدة .

أ- تعمل علامة الضرب (*) بالتحقق من أن الحرف الذي توضع بعد متكرر مرة أو أكثر أو غير موجوبتاتاً ,مثلاً لاحظ هذه الكلمة **Bea*t** فإن وضعناها في البحث فأنها ستجد أي من هذه الكلمات **(Beaat , Beat , Bet)** .

ب- تعمل علامة الجمع (**+**) بالتحقق من وجود العنصر الذي يسبقها مرة واحدة أو أكثر ,مثلاً لاحظ هذه الكلمة **Bea+t** إن وضعناها في البحث فإنها ستجد أي من هذه الكلمات **(Beat , Beaat)** .

ت- تعمل علامة الاستفهام (**?**) بالتأكد من وجود العنصر الذي يسبقها مرة واحدة أو عدم وجوده بتاتاً ,مثلاً لاحظ هذه الكلمة **Bea?t** إن وضعناها في البحث فإنها ستجد أي من هذه الكلمات (Beat , Bet) .

* لاحظ أن هذه العلامات الثلاثة في هذه الأمثلة السابقة كانت مسبوقة بحرف واحد فقط , وإن أردنا أن يشمل تأثيرها على عدة أحرف قبلها فإننا نضع تلك الأحرف في قوسين ( ) ,مثلاً :-

(wo)?man

ملاحظة/ يمكننا التأكد من تكرار حرف بشكل معين من المرات أو أكبر من عدد معين من المرات أو أصغر من عدد معين من المرات فيمكننا ذلك باستخدام القوسين { x , y } ,مثلاً لو أردنا أن نتأكد من أن الحرف d مكرر مرتين إلى أربع مرات سنكتب d{2,4} , أما إذا أردنا أن نتأكد من أنه مكرر أكث من مرتين إلى عدد غير محدد من المرات سنكتب d{2, } , أما إذا أردنا أن يتكرر كحد أعلا 4 مرات سنكتب d{ ,4} و وإذا أردنا أن يتكرر بعدد معين من المرات سنكتب d{8} .

* الاختصار **\b** معناه أي شيء ولكن ليسحرفاً (الحروف ألتي بين \w و \W تقريباً) , إليك هذا الجدول أذي هو ملخص لما أخذناه من قواعد :-

القاعدة	المعنى
[a b c]	أي حرف كان a أو b أو c
[^ a b c]	أي ما عدا حرف a أو b أو c
\b	الحروف التي بين \w و \W
.	أي حرف
(abc)	تعتبر abc كمجموعة
?	حرف أو مجموعة حروف مكررة مرة أو غير مكررا هائياً

+	**حرف أو مجموعة حروف مكررة مرة أو أكثر**
*	حرف أو مجموعة حروف مكررة مرة أو أكثر أو غير مكررة هائياً
{ x, y }	تكرار محصور بين عددين
{ , y }	تكرار بحد أقصى من المرات
{ x , }	تكرار بحد أدنا من المرات
{ x }	تكرار بعدد معين
^	في بداية النص
$	في نهاية النص

مثال :- للتأكد من إميل

```
^[-a-z A-Z 0-9-]+(\.[-A-Z a-z 0-9-]+)*@[a-z A-Z 0-9-]+(\.[a-z A-Z 0-9-]+)*$
```

شرح التعبير :-

الرمز	الشرح
^	يجب أن يبدأ النص
[-a-z A-Z 0-9-]	أي حرف من a إلى z كبير أو صغير أو أرقام
+	وقد يكون هذا الحرف مكرر أكثر من مرة
(\.[-a-z A-Z 0-9-]+)	بالإضافة إلى انه قد يتبع بالنقطة أو حروف و أرقام
*	وقد لا يتبعه أو يتبعه ويتكرر أكثر من مرة
@	وبعد ذلك يكون لديه الحرف @
*([a-z A-Z 0-9-]+(\.[a-z A-Z 0-9-]+)	أيضاً نفس القواعد في النهاية
$	يجب أن ينتهي النص

مثال :-

```
function mer( $mail , $t) {
Sx = "^[-a-z A-Z 0-9-]+(\.[-A-Z a-z 0-9-]+)*@[a-z A-Z 0-9-]+(\.[a-z A-Z 0-9-]+)*$" ;
if (ereg ($x , $mail) { $r = " the mail is true " ;
echo $r ; } else {$r = "the mail is not true" ;
echo $r ; } return ;}
mer ("ahmedcia71@yahoo.com" , $t) ;
```

- الدالة eregi ( ) ;

وهي مثل الدالة ereg ولكن الفرق بينهما إن هذه الدالة لا تفرق بين الحروف الكبيرة والصغيرة .

- الدالة ereg_replace

نستطيع باستخدام هذه الدالة التخلص من بعض الحروف أو استبدالها بحروف أخرى , والصيغة العامة لها هي :-

ereg_replace (اسم المتغير الذي سيجري عليه التغيير , " الحرف الجديد ", "الحرف القديم") ;

مثال :-

```
$x = "C:\windows\desktop" ;
$y = "Ail love his game .... " ;
$newx = ereg_replace ("[\.]" , "/" , $x ) ;
$newy = ereg_replace ("\." , " " , $y ) ;
echo $newx ;
echo "<br>" ;
echo $newy ;
```

- الدالة preg_match

تستخدم هذه الدالة بالبحث عن نمط للتعابير النظامية في داخل السلسلة النصية , في حالة وجود مطابقة تعيد القيمة true وتعيد القيمة false بخلاف ذلك .

مثال1/

```
<?php
$reg = '/(\d{4})-(\d{1,2})-(\d{1,2})/';
$date1 = '1995-5-21';
$date2 = '95-May-21';
if(preg_match($reg, $date1) != false)
{
echo "Date '$date1' is a valid Date";
} else {
echo "Date '$date1' is a NOT valid Date";
```

```
}
echo '<br>';
if(preg_match($reg, $date2) != false)
{
echo "Date '$date2' is a valid Date";
} else {
echo "Date '$date2' is a NOT valid Date";
}
?>
```

مثال2/

```
<?php
$reg = ' (\d{4})-(\d{1,2})-(\d{1,2}) ';
$date = '1995-5-21';
if(preg_match($reg, $date, $results) != false)
{
echo "Date '$date' is a valid Date";
echo '<br>';
echo "The full match is {$results[0]} <br>";
echo "The Year is {$results[1]} <br>";
echo "The Month is {$results[2]} <br>";
echo "The Day is {$results[3]}";
}
else
{
echo "Date '$date' is a NOT valid Date";
}
?>
```

- **الدالة preg_repalce**

تقوم هذه الدالة باستبدال نص بنص آخر بالاعتماد على التعابير النظامية .

مثال/ يحول صيغة التاريخ من الشكل "YYYY-MM-DD" إلى الشكل "DD | MM | YYYY"

```
<?php
$x = ' (\d{4})-(\d{1,2})-(\d{1,2}) ' ;
$y = ' $3 | $2 | $1 ' ;
echo preg_replace($x , $y , '1995-5-21') ;
?>
```

- **الدالة filter_var**

تستخدم هذه الدالة للتحقق من المتغير وذلك بشكل جاهز وهي تأخذ بارامترين الأول هو المتغير والثاني هو نوع التحقق أي التحقق الذي نريده مثلا هل المتغير يحمل قيمة لبريد الكتروني او يحمل ip او يحمل قيمة رابط .

مثال/

```
$x = " ahmed@gmail.com " ;

if( filter_var( $x , FILTER_VALIDATE_EMAIL ) ){

echo " yes " ;

} else {

echo " no " ; }
```

كما تلاحظ هنا تحققنا من أن المتغير يحمل بريد اليكتروني , وهناك انواع أخرى من التحققات منها :-

1- FILTER_VALIDATE_BOOLEAN للتحقق من القيمة البولية (1 , yes , true) .
2- FILTER_VALIDATE_EMAIL للتحقق من الاميل .
3- FILTER_VALIDATE_FLOAT للتحقق من الكسور العشرية .
4- FILTER_VALIDATE_INT للتحقق من القيم الرقمية الصحيحة .
5- FILTER_VALIDATE_IP للتحقق من الـ ip .
6- FILTER_VALIDATE_URL للتحقق من الرابط .

- **الدالة ctype_alnum**

تستخدم هذه الدالة للتحقق من ان النص الممرر لها يتكون فقط من حروف كبيرة او صغير او ارقام , ولاحظ بأنه اذا وجد فاصلة (مسافة فارغة) بين النص ستعتبره حرف خاطئ وتعيد القيمة false اما اذا كان كل شيء صحيح تعيد القيمة true

مثال/

```
$val = "Welcome34";
if( ctype_alnum( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_alpha**

تستخدم هذه الدالة للتحقق من ان النص الممرر لها هو حروف فقط (كبيرة او صغير) واذا كان يحتوي على أي شيء آخر مثل فاصلة او رمز او رقم ستعيد القيمة false وبخلاف ذلك ستعيد القيمة true

مثال/

```
$val = "Welcome";
if( ctype_alpha( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_lower**

تستخدم هذه الدالة للتحقق من ان النص الممرر لها هو حروف صغير فقط واذا كان يحتوي على أي شيء آخر مثل حرف كبير او فاصلة او رمز او رقم ستعيد القيمة false وبخلاف ذلك ستعيد القيمة true

مثال/

```
$val = "Welcome";
if( ctype_lower( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_upper**

تستخدم هذه الدالة للتحقق من ان النص الممرر لها هو حروف كبيرة فقط واذا كان يحتوي على أي شيء آخر مثل حرف صغير او فاصلة او رمز او رقم ستعيد القيمة false وبخلاف ذلك ستعيد القيمة true

مثال/

```
$val = "WELCOME";
if( ctype_upper( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_digit**

تستخدم هذه الدالة للتحقق من ان النص الممرر لها هو أرقام فقط واذا احتوى على أي شيء آخر ستعيد القيمة false

مثال/

```
$val = "50273";
if( ctype_alpha( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_cntrl**

تستخدم هذه الدالة للتحقق إذا كان النص الممرر لها يتكون فقط من الحروف السلاشية control

ملاحظة/ الحروف السلاشية هي مثل \f \r \b \n \t .... إلخ

مثال/

```
$val = "\t\n";
if( ctype_cntrl( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_punct**

تستخدم هذه الدالة للتحقق من ان النص المرر لها يتكون فقط من الرموز punctuation أي لا يحتوي على ارقام ولا على حروف ولا على فاصلة .

مثال/

```
$val = "#%@{)";
if( ctype_punct( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

- **الدالة ctype_space**

تستخدم هذه الدالة للتحقق من ان النص الممرر لها يتكون من المسافات فقط (نقصد بالمسافات هو كل فراغ سواء نتج عن الضغط على المسطرة او تاب او أنتر من لوحة المفاتيح) ستعيد القيمة true اذا كان يتكون من المسافات فقط وfalse بخلاف ذلك .

مثال/

```
$val = "  \n   \t          ";
if( ctype_space( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

لاحظ بأن \n يعادل الضغط على مفتاح enter والرمز \t يعادل الضغط على مفتاح tab .

- **الدالة ctype_xdigit**

تستخدم هذه الدالة للتأكد من ان النص الممرر لها هو فقط يتكون من حروف وارقام النظام الستعشري , ستعيد القيمة true اذا كان كذلك وبخلافة تعيد القيمة false .

ملاحظة/ يقصد بالنظام الستعشري هي هذه الأرقام والحروف فقط (123456789abcdef) ويستفاد من هذا النظام في أشياء كثيرة منها مثلاً القيم اللونية .

مثال/

```
$val = "52cf";
if( ctype_xdigit( $val ) ) {
echo "Yes";
} else {
echo "No"; }
```

تجاهل الأخطاء

إن وضعت خطأمتعمداً وأردت تجاهل هذا الخطأ فيمكن ذلك بوضع العلامة @ قبل الدالة التي فيها خطأ , أنت تعلم إن قسمة رقم على صفر يعتبر خطأ في لغة php, فمثلاً لو أردت تجاهل هذا الخطأ وعدم طباعة الخطأ والاستمرار في البرنامج , لاحظ :-

```
function bob($y) {
$y = $y / 0 ;
return $y ; }
$x = @bob(44) ;
echo $x ;
```

الفصل السادس عشر

(session & cookies)

أولاً :- cookies

وهي ملفات يتم تخزينها داخل جهاز المستخدم للتعرف عليه عند زيارة موقعنا مرة أخرى , لزراعة ملف الكوكيز نحتاج إلى الدالة setcookie والصيغة العامة لها هي :-

setcookie (اسم المتغير , قيمة المتغير , تاريخ نهاية الكوكيز) ;

مثال :-

```
setcookie ("myname" , "MyValue" , time()+(3600) ) ;
```

ويمكن أن نضيف قيم أخرى بين قوسي الدالة setcookie بالإضافة إلى القيم الثلاثة السابقة وهذه القيم اختيارية أي يمكن عدم وضعها (نفصل بين قيمة وأخرى بفارزة) ومن هذه القيم مثلا يمكن إضافة العلامة سلاش " / " بعد قيمة الوقت وهذه العلامة تعني أن هذا الكوكيز (الكعكة) ستكون متاحة لجميع مجلدات الموقع لكن إذا أضفنا مع هذه العلامة اسم المجلد مثلا " /example " هنا سيكون هذا الكوكيز متاح لهذا المجلد فقط , وهنا قيمة أخرى يمكن إضافتها بعد القيمة السابقة وهي تأخذ بشكل افتراضي القيمة false لكن يمكن أن نعطيها القيمة true حيث إذا كانت هذه القيمة true فهذا يعني أن الكعكة يجب أن يتم نقلها بواسطة اتصال آمن عن طريق HTTPS , وهناك قيمة أخرى أيضاً يمكن إضافتها بعد القيمة السابقة وهي تشير إلى أن الكعكة لا يمكن الوصول إليها إلا عن طريق بروتوكول HTTP وهذا يعني أن القيم المخزنة في الكعكة لا يكمن الوصول إليها عن طريق JavaScript على سبيل المثال .

ملاحظة/ لايصح أن نعرف الكوكيز بعد أي مخرجات لذلك يفضل كتابته في بداية الصفحة.

- **الدالة $_COOKIE**

هذه الدالة هي التي يتم تخزين بيانات الكعكات داخلها على شكل مصفوفة ومن خلالها يمكن عرض البيانات المخزنة .

مثال/

```
echo $_COOKIE['myname'];
```

حيث سيتم طباعة القيمة المخزنة في المتغير myname وهو اسم المتغير الذي خزنا القيمة داخله مسبقاً من خلال الدالة setcookie

مسح الكوكيز

يمكننا مسح الكوكيز بأكثر من طريقة ، بالطبع فإن المستخدم يستطيع مسح الكوكيز وتغيير محتوياتها بنفسه ولكن في حالة ما إذا أردنا أن نجعل السيرفر يقوم بمسحها فإننا نستخدم إحدى هاتين الطريقتين

أما أن نقوم بإخبار السيرفر بوقت قديم :

```
setcookie ("ahmed" , " 0 " , time() - 999) ;
```

وإما القيام بمسح الكوكيز بكتابة اسمه فقط :

```
setcookie ("ahmed") ;
```

ثانياً :- الجلسات sessions

الجلسة هي آلية لتتبع المستخدم وهو يقوم بمختلف العمليات داخل الموقع حيث يتم تخزين هذه البيانات على جهاز السيرفرعوضاً عن حفظها على جهاز المستخدم كما هو الحال في الكوكيز , لكل مستخدم id خاص به يسمى session id او اختصارا sid .

لبدأ الجلسة يجب تضمين session_start( ); في رأس كل صفحة نود استخدام الجلسات فيها .

- الدالة $_SESSION

وهي عبارة عن مصفوفة يتم تخزين بيانات الجلسات فيها ومن خلالها يتم إضافة متغير خاص بالجلسة مباشرة , وتكون الصيغة العامة لها بهذا الشكل

```
$_SESSION[' var '] = value;
```

لاحظ أن var يمثل اسم المتغير الذي ستخزن فيه القيمة ويمكن اختيار أي اسم , ويمكن طباعته بهذا الشكل :-

```
echo  $_SESSION[‘ var ‘] ;
```

ولحذف متغير خاص بالجلسات يمكن إسناد قيمة فارغة له .

- الدالة unset

تستخدم هذه الدالة لحذف متغير خاص بالجلسات , مثال/

```
unset($_SESSION['name']);
```

- الدالة session_destroy( );

تستخدم هذه الدالة لحذف كامل الجلسة وذلك بكتابتها فقط مع ملاحظة بأنه يجب أن نكتب قبلها الدالة session_start .

مثال/

```
session_start( ) ;

session_destroy( ) ;
```

الفصل السابع عشر

(MySQL)

تقسم MySQL إلى جزأين هما :-

1- لغة تعريف البيانات DDL

وتعمل هذه اللغة على إنشاء وتعديل قواعد البيانات والجداول , وأهم أوامرها create table , alter , create database .

2- لغة معالجة البيانات DML

وهذه اللغة تختص بالتعامل مع البيانات المدرجة داخل الجداول من عمليات حذف وتحرير وإدراج وتعديل ... الخ , وأهم أوامرها update , insert , delete , replace , select .

ملاحظة/ بعد كتابة أي سطر في MySQL يجب وضع فارزة منقوطة (;) في نهايته ثم نضغط enter , ولاحظ أنه إن كان الكود يتكون من أكثر من سطر يجب أن نكتب بشكل أفقي وفي نهاية السطر هو سينزل إلى السطر التاليتلقائياً ولا يمكن أن ننزل نحن إلى السطر التالي بالضغط على enter لأن الكود سيكون مجزأ ولن تتعرف MySQL عليه .

* عند الدخول في MySQL يمكن الخروج من خلال كتابة exit ثم enter (لاحظ عدم وضع فارزة منقوطة (;) بعد exit) .

* لكي نظهر جميع قواعد البيانات التي لدينا نكتب الأمر التالي :-

```
show databases ;
```

سوف تشاهد جميع قواعد البيانات التي لديك , والآن لكي نتعامل مع قاعدة بيانات معينة يجب أن نخبر MySQL عنها , ولنفترض إننا نريد أن نتعامل مع قاعدة البيانات المسماة mysql فيجب علينا كتابة الأمر التالي :-

```
use mysql ;
```

والآن يتوقع البرنامج إن أي استفسار قادم له سوف يكون موجه إلى هذه القاعدة , والآن لكي نشاهد الجداول الموجودة في قاعدة البيانات هذه نحتاج إلى كتابة الأمر التالي :-

```
shoe tables ;
```

أو نستطيع كتابة الأمر بهذا الشكل (إن لم نكن قد حددنا قاعدة البيانات التي سوف نتعامل معها سابقاً)

```
show tables from mysql ;
```

اسم قاعدة البيانات

ستلاحظ ظهور الجداول الموجودة في هذه القاعدة , ولكي نتعرف على الحقول الموجودة ونوعها والشروط المفروضة على البيانات التي سوف تسجل بها نكتب الأمر التالي :-

```
show  columns  from  user ;
```

اسم الجدول الموجود في قاعدة البيانات

أو يمكن كتابة الأمر بهذا الشكل حيث لا يوجد فرق بين الأمرين :-

```
describe user ;
```

ولكي نتعرف على جميع البيانات الموجودة في الجدول المسمى user نكتب الأمر التالي :-

```
select * from user ;
```

عند تنفيذ هذا الأمر سوف تلاحظ إن البيانات التي حصلت عليها كثيرة ولا يمكن قراءتها ولكي نحصل على بيانات مرتبة وواضحة يجب أن نطلب بيانات أقلفمثلاً يمكننا أن نطلب فقط الأسماء وكلمات السر الموجودة في هذا الجدول وذلك بكتابة الأمر التالي :-

```
selec t user , password  from  user ;
```

أسماء حقول موجودة في الجدول اسم الجدول

لاحظ أننا حصلنا على جميع البيانات الموجودة في هذين الحقلين , ولكن ماذا لو كنا نريد فقط البيانات الخاصة باسم مستخدم واحد المسمى root , سوف نكتب الأمر التالي :-

```
select user , password from user where user = 'root' ;
```

* الآن سوف نتعرف كيف يمكننا أن ننشأ قاعدة بيانات ,فمثلاً لو أردنا أن ننشئ قاعدة بيانات ونسميها school سنكتب الأمر التالي :-

```
create database school ;
```

سوف نشاهد رسالة تؤكد إن قاعدة البيانات تم إنشائها .

ولو أردنا أن نمسح قاعدة البيانات هذه (school) فسوف نكتب الأمر التالي :-

```
drop database school ;
```

ملاحظة/ في MySQL لا فرق بين الحروف الكبيرة والصغير بالنسبة إلى الدوال , مثلا :-

SELECT = select

ملاحظة/ MySQL لا تهتم بالفواصل إن كانت فاصلة واحدة أو أكثر بالنسبة إلى الدوال ولكن لا يمكن أن لا نضع أي فاصلة بين الكلمتين .

الآن لو أردنا أن ننشئ جدول اسمه std فيه الحقول التالية (الرقم والاسم والدرجة) , سوف نكتب الكود التالي :-

ولكن في البداية نخبر السيرفر على إننا نستخدم قاعدة البيانات school بواسطة هذا الأمر :-

```
use school ;
```

ثم نكتب الأمر التالي :-

```
create table std (no int(5) not null , name varchar(20) not null , gride int(3) not null)
;
```

عند تنفيذ الكود السابق سوف تشاهد رسالة تخبرك بنجاح العملية .

سأشرح الآن الكود السابق :-

كما تعرف فأن (create table) تخبر السيرفر عن إنشاء جدول جديد وقد سمينا هذا الجدول (std) بعد ذلك نفتحقوساً لنضع حقول الجدول حيث وضعنا na وهي تمثل الرقم (number) ويمكن وضع أي اسم آخر , بعد ذلك وضعنا int وهي تمثل القيم العددية حيث إن وضعنا int في أي حقل سيدل ذلك على إن هذا الحقل لن يستقبل إلا الأعداد , وبعدها وضعنا بين قوسين الرقم (5) حيث إن هذا يدل على إن أكبر قدر يمكن أن يحتوي هذا الحقل هو خمس أرقام وبعدها وضعنا (not null) وهذه تشير إلى أن هذا الحقل لا يمكن أن يتركفارغاً وبعدها وضعنا فارزة لنبين

انتهاء هذا الحقل ونعرف حقل جديد , ولاحظ إننا هنا عرفنا ثلاثة حقول في هذا الجدول . حيث إن (varcher) تشير إلى أن قيم هذا الحقل هي فقط حروف .

* لو أردنا حذف الجدول السابق نكتب الأمر التالي :-

```
drop table std ;
```

وعند تنفيذ ذلك سوف تشاهد رسالة تخبرك عن نجاح العملية .

ولو أردنا أن نضيف بيانات في حقول الجدول السابق سوف نكتب الأمر التالي :-

```
insert into std (no,name,gride) value (1,'sager',95) ;
```

وعند تنفيذ ذلك سوف تشاهد رسالة تخبرك عن نجاح العملية .

ملاحظة/ عند كتابة قيمة نصية نضعها بين علامتي تنصيص مفردة (' ') وأن وضعنا قيمة عددية فلا داعي لعلامات التنصيص .

الفصل الثامن عشر

(تعليمات SQL)

ملاحظة مهمة/ سنفترض أننا أنشأنا جدول للطلاب اسمه std وفيه ثلاثة حقول وأسمائها :- رقم الطالب no واسم الطالب name ودرجة الطالب gride . وسنجري الأمثلة القادمة على هذا الأساس .

- **التعليمة select**

تستخدم التعليمة select لاسترجاع البيانات من جدول أو جداول في قاعدة البيانات , والصيغة العامة لها هي :-

```
select col1 , col2 , ... from table1 , table2 , ... ;
```

أسماء الحقول أسماء الجدول

مثال :-

```
select no , name from std ;
```

لاحظ هنا في هذا المثال سوف يعيد لنا جميع أرقام وأسماء الطلبة الموجودين في هذا الجدول .

* لو أردنا استرجاع جميع الحقول وجميع السجلات في هذا الجدول سنكتب الكود التالي :-

```
select * from std ;
```

- **التعليمة where**

وتستخدم هذه التعليمة لاستخراج بيانات محددة من الجدول التي سينطبق عليها الشرط , والصيغة العامة لها هي :-

```
select col1 , col2 , ... from table1 , table2 , ... where col = ??? ;
```

مثال :- لو أردنا الحصول على رقم واسم ودرجة الطالب من جدول الطلبة (std) للطالب الذي رقمه 4 فسوف نكتب الكود التالي :-

```
select no , name , gride from std where no = 4 ;
```

حيث هنا أخبرنا السيرفر بأننا نريد رقم واسم ودرجة الطالب عندما يتحقق الشرط وهو أن يكون رقمه يساوي 4 ولكن إن أردنا أن نبحث من خلال اسم الطالب فسوف يكون الكود كالتالي :-

```
select * from std where name = 'Ahmed' ;
```

* لا يشترط أن تكون المقارنة فقط باستخدام علامة المساواة حيث يمكن استخدام العلامات الأخرى مثل

(=< , => , <> , > , <) .

فمثلاً لمشاهدة الطلبة الذين درجاتهم أكبر من 90 سنكتب هذا الكود :-

```
select * from std where gride > 90 ;
```

أ- العلامة like

تتيح لنا هذه العلامة المقارنة بجزء من المعلومات, مثلاً لو أردنا أن نتعرف على الطلبة الذين تبدأ أسمائهم بحرف s سنكتب الجملة التالية :-

```
select * from std where name like 's%' ;
```

ولو أردنا التعرف على الطلبة الذين يوجد من ضمن أسمائهم الحرف (s) (في أي مكان من الاسم) سيكون الكود كالتالي :-

```
select * from std where name like '%s%' ;
```

لاحظ لو أردنا أن نتعرف على الاسم الذي ينتهي بحرف s مثلاً , نضع '%s' ومن ذلك نستنتج أن العلامة % تمثل حرفاً (أي حرف) .

ب- العلامة in

يمكن تعريف هذه العلامة على إنها الشرط (توفر أحد هذه القيم) أي ان الشرط يتحقق إذا كان أحد هذه القيم صحيح .

مثال :- لو أردنا درجات الطلبة الذين أرقامهم 1 و 3 و 4 فإننا سوف نكتب الجملة التالية :-

```
select * from std where no in (1,3,4) ;
```

ج- العلامة between

بهذه العلامة نخبر السيرفر إننا نحتاج إلى القيم الواقعة بين هاتين القيمتين , وتستخدمعادتاً في البحث بين تاريخين معينين .

مثال :- عندما نحتاج إلى الطلبة التي درجاتهم بين 80 و 90 نكتب الجملة التالية :-

```
select * from std where gride between 80 and 90 ;
```

* لعرض الطلبة التي درجاتهم لا تكون من ضمن الدرجات المحصورة بين 80 و 90 نضع not قبل between .

* عند استخدام التواريخ نضع العلامة # بدل من علامة التنصيص (' ') .

دمج الشرط

نستطيع كتابة أكثر من شرط على نفس الحقل أو حقول متعددة . ولكي نفعل ذلك نستخدم الكلمتين التاليتين :-

and :- يجب تحقق الشرطين .

or :- تحقق أحد الشرطين .

مثال :- لو أردنا أن نعرف الطلبة الذين أرقامهم أكثر من 2 ودرجاتهم أكبر من 70 نكتب الجملة التالية :-

```
select * from std where no > 2 and gride > 70 ;
```

ويمكن أن نضع or مكان and .

- **التعليمة order by**

نحتاج في كثير من الأحيان إلى ترتيب البيانات حسب صف معين , ولذلك نستخدم هذه التعليمة , وتأتي مع هذه التعليمة الكلمتان :-

Asc :- ترتيب الحقل من الأصغر إلى الأكبر (وهو الافتراضي) .

Desc :- ترتيب الحقل من الأكبر إلى الأصغر .

مثال :- للحصول على البيانات مرتب حسب الدرجة من الأصغر إلى الأكبر نكتب الجملة كالتالي :-

```
select * from std order by gride asc ;
```

وإن أردنا الترتيب بالعكس نضع (desc) مكان (asc) , وإذا أردنا ترتيب البيانات بحسب الاسم والدرجةمثلاً , سوف نكتب الكود التالي :-

```
select name , gride from std order by name , gride ;
```

- **التعليمة limit**

وهذه التعليمة تحدد عدد السجلات الأقصى الذي نحصل عليه , حيثمثلاً لو أردنا استرجاع أول ثلاث سجلات فقط سنكتب الجملة التالية :-

```
select * from std limit 3 ;
```

مثال :- لو أردنا أن نحضر السجلين الرابع والخامس , سوف نكتب الجملة التالية :-

```
select * from std 3 , 2 ;
```

حيث إن الرقم الأول يمثل الرقم الذي نبدأ العد بعده والرقم الثاني يمثل عدد السجلات .

- **التعليمة insert into**

نستخدم هذه التعليمة عندما نحتاج إلى إضافة بيانات إلى الجدول .

مثال :- لو أردنا إضافة بيانات طالب جديد اسمه الأول yosef ودرجته 92 ورقمه 6 , نحتاج إلى كتابة هذه الجملة :-

```
insert into std (no,name,gride) values (6,'yosef',92) ;
```

وعند تنفيذ هذه الجملة تظهر رسالة تؤكد أنه تم إضافة سجل واحد إلى الجدول ,أيضاً نستطيع كتابة الجملة السابقة بهذا الشكل :-

```
insert into std values (6,'yosef',92) ;
```

ولكن يجب أن نكتب الحقول بالترتيب حسب ترتيبها في الجدول .

* وكذلك يمكن اضافة البيانات إلى الحقول بهذه الطريقة , لاحظ :-

```
Insert  into  std  set no=6, name='yosef',gride=92 ;
```

- **التعليمة update**

تستخدم هذه التعليمة لتعديل بيانات محددة في الجدول .

مثال :- لو إن الطالب رقم 3 أوضح إن اسمه هو Ali وليس Ahmed وطلب منا تعديل ذلك , سوف نستخدم الجملة التالية :-

```
update std set name = 'Ali' where no = 3 ;
```

لاحظ هنا حيث استخدمنا التعليمة where وإن لم نستخدمها فسوف تحول جميع الأسماء الموجودة في الجدول إلى Ali , ولاحظأيضاً إن أردنامثلاً تغيير الاسم والدرجةمعاً , سنكتب الجملة التالية :-

```
update std set name = 'Ali' , gride = 90 where no = 3 ;
```

- **التعليمة delete**

عندما نحتاج حذف سجل من الجدول نستخدم هذه التعليمة .

مثال :- لو أردنا حذف سجل الطالب رقم 4 من الجدول , سنكتب الجملة التالية :-

```
delete from std where no = 4 ;
```

وعند تنفيذ هذه الجملة سوف نحصل على رسالة تبين إن العملية تمت بنجاح , لاحظ هنا في هذا المثال وضعنا التعليمة where وإن لم نضعها فسوف تحذف جميع السجلات .

- **التعليمة count**

للحصول على عدد السجلات من حقل معين .

مثال:- لتتعرف على عدد الطلبة الموجودين لدينا في قاعدة البيانات نكتب الجملة التالية :-

```
select  count(*)  from std ;
```

لاحظ عدم وجود فاصلة

- **التعليمة max**

للحصول على أكبر قيمة في السجلات .

مثال :- للحصول على أكبر درجة حصل عليها طالب نكتب الجملة التالية :-

```
select max (gride) frome std ;
```

- **التعليمة min**

للحصول على أقل قيمة في السجلات .

مثال :- للحصول على أكبر درجة حصل عليها طالب نكتب الجملة التالية :-

```
select min (gride) from std ;
```

- **التعليمة avg**

للحصول على معدل السجلات .

مثال :- للحصول على معدل الدرجات التي حصل عليها طالب نكتب الجملة التالية :-

```
select avg (gride) from std ;
```

- **التعليمة sum**

للحصول على مجموع البيانات .

مثال :- لنتعرف على مجموع الدرجات التي حصل عليها الطلبة نكتب الجملة التالية :-

```
select sum (gride) from std ;
```

- **التعليمة distinct**

تعمل هذا التعليمة على منع التكرار .

مثال :- لعرض أسماء الطلاب ولكن بدون أن نعرض الأسماء المكررة نكتب الجملة :-

```
select distinct name from std ;
```

- **التعليمة inner join**

تستخدم هذه التعليمة لربط جدولين مع بعضهما البعض أي سيمكننا أن نستخرج البيانات من كلا الجدولين في نفس الوقت وكأنهما جدول واحد مع ملاحظة أن الجدولين لن يتم دمجهما في جدول واحد في قاعدة البيانات لكن لغة الـ php ستتعامل معهم كجدول واحد , ولاحظ أن الجدولين يجب أن يحتوي كل منهم على حقل يكون هذا الحقل مضاف في كليهما .

مثال /

```
mysql_query(“ select * from std inner join std2 on std.id=std2.id “);
```

حث هنا الحقل المشترك بين الجدولين هو id , ولاحظ بأنه يمكن أن نكتب بعد هذه الجملة أي تعليمة نشاء مثل التعليمة where والتعليمة order by وغيرها من التعليمات بشكل أعتيادي .

- **التعليمة truncate table**

تستخدم هذه التعليمة لإفراغ محتوى جدول أي تجعله فارغ تماماً من أي بيانات تمت إضافتها إليه أي وكأنه قد تم إنشائة للتو أي ستبقى فقط أسماء عناوين الحقول الأصلية .

مثال/

```
truncate table std ;
```

- **الأمر alter table**

ويستخدم في التعديل على الجدول من إضافة وحذف أعمدة (حقول) .

* لإضافة حقل تكون الصيغة كالتالي :-

```
alter table اسم الجدول add اسم الحقل ( نوع البيانات int أو varchar أو ...الخ)
```

* لحذف حقل تكون الصيغة كالتالي :-

```
alter table اسم الجدول drop column اسم الحقل
```

الفصل التاسع عشر

(دوال php للتعامل مع MYSQL)

* لربط صفحة php بقاعدة بيانات نحتاج إلى خطوتين الخطوة الأولى من خلال إجراء الاتصال من خلال الدالة التالية :-

الخطوة الأولى :-

- **الدالة mysql_connect**

هذه الدالة تستخدم لأجراء الاتصال بقاعدة البيانات و هي تحتاج إلى ثلاثة متغيرات :-

1- اسم السيرفر .
2- اسم المستخدم .
3- كلمة السر .

وتعود هذه الدالة بمقبض الاتصال في حال نجاح الاتصال أو رسالة تعبر عن الخطأ في حال فشل الاتصال .

مثال :-

اسم السيرفر كلمة السر

```
<?php
$link = mysql_connect ("localhost" , "root" , " ")
or die ("فشل الاتصال") ;
print ("تم الاتصال") ;

mysql_close ($link);
?>
```

اسم المستخدم

في هذا المثال عند حدوث الاتصال سوف يتم طباعة العبارة تم الاتصال , ولكن عند فشل الاتصال ستطبع العبارة فشل الاتصال وتلاحظ بأننا قمنا بإنهاء الاتصال بواسطة الدالة **mysql_close** .

ملاحظة/ يمكن اجراء الأتصال من خلال الدالة mysql_pconnect و هي تكتب بنفس طريقة الدالة السابقة والفرق بينهما هو أن الدالة السابقة يجب أغلاقها بواسطة الدالة mysql_close بعد الأنتهاء تماماً من الكود أما هذه الدالة فلا تحتاج إلى الأغلاق لكنها سوف تعمل أتصال جديد

لكل مستخدم يدخل إلى الموقع وهذا سيعمل ضغط على الخادم لذا يفضل استخدام الدالة السابقة mysql_connect .

الخطوة الثانية :-

نحدد قاعدة البيانات التي نود التعامل معها من خلال الدالة التالية :-

- **الدالة mysql_select_db**

تحدد هذه الدالة قاعدة البيانات التي نتعامل معهاحالياً , وتحتاج هذه الدالة متغيرين هما :-

1- اسم قاعدة البيانات التي نود التعامل معها .

2- مقبض الاتصال الذي أجريناه .

وتعود هذه الدالة بالقيمة true في حال نجاحها , أو false في حال حدوث خطأ .

مثال :-

```
<?php
$link = mysql_connect ("localhost" , "root" , "  ")
or die ("فشل الاتصال") ;
print ("تم الاتصال") ;
mysql_select_db ("school" , $link) ;
?>
```

وبذلك نستطيع الحصول أو التحكم بجميع الجداول الموجودة بقاعدة البيانات ولكي نفعل ذلك توفر لنا php عدة دوال سوف نشرحها فيما يلي .

دوال التعامل مع قواعد البيانات

- **الدالة mysql_query**

تنفذ هذه الدالة جمل الاستعلام على قواعد البيانات وتعود لنا بالقيم الناتجة من عملية الاستعلام على شكل مقبض لعملية الاستعلام .

مثال :-

```
$result = mysql_query ("select * from std") ;
```

تلاحظ أن القيم المستخرجة وضعت في المتغير result ولكي نستفيد من هذه القيم نستخدم إحدا هذه الدوال الموضحة في الأسفل :-

1- الدالة mysql_fetch_row

تضع هذه الدالة القيم المستخرجة في مصفوفة لكي نتعامل معها بسهولة ونحتاج فقط إلى مقبض عملية الاستعلام .

مثال :-

```
$row = mysql_fetch_row ($result) ;
```

الآن أصبح لدينا مصفوفة نستطيع التعامل معها بكل سهولة لاحظ :-

```
$x = $row['id_cat'];
```

اسم الحقل في قاعدة البيانات

* يمكن وضع array مكان row حيث إن row سيخرج الصف فقط أما array فسوف تستخرج المصفوفة , أي يمكن كتابة الدالة السابقة بهذا الشكل (**mysql_fetch_array**) .

2- الدالة mysql_fetch_assoc

تستخدم هذه الدالة لاستخراج البيانات من الجدول في قاعدة البيانات على شكل مصفوفة وهي نفس الدالة السابقة تماماً .

مثال/

```
$row = mysql_fetch_assoc ($result) ;

$x = $row['id_cat'];
```

اسم الحقل في قاعدة البيانات

3- الدالة mysql_fetch_object

تستخرج هذه الدالة القيم من قاعدة البيانات ويتم التعامل معها ككائن ونحتاج فقط إلى مقبض عملية الاستعلام , وهي مشابهه للدالة السابقة , ولاحظ أنه يتم التعامل مع القيم المستخرجة بهذا الشكل :-

مثال :-

```
$row = mysql_fetch_object($result) ;

$x = $row -> id_cat ;
```

اسم الحقل في قاعدة البيانات

- **الدالة mysql_free_result**

تستخدم هذه الدالة بعد الانتهاء من عملية الاستعلام بواسطة الدالة mysql_query حيث أنها تعتبر بمثابة إغلاق للأستعلام وهي مهمة لكي لا يكون هناك ضغط على الخادم خصوصاً إذا كانت هناك بيانات كثيرة , وهي عادتاً تستخدم إذا كنا قد استخدمنا select وبعدها حلقة تكرار .

مثال/

```
$result = mysql_query ("select * from std") ;

$row = mysql_fetch_object($result) ;

$x = $row -> id_cat ;

mysql_free_result( $result );
```

- **الدالة mysql_num_rows**

تعود لنا هذه الدالة بعدد السجلات التي حصلنا عليها بعد استخدام التعليمة select فقط , وتحتاج هذه الدالة إلى متغير واحد فقط وهو مقبض الاتصال .

ملاحظة/ لا تستخدم هذه الدالة مع التعليمات (update , insert , delet) .

* ويمكن وضع fields مكان rows لتأتي لنا بعدد الفيلات وتكون الدالة بهذا الشكل (**mysql_num_fields**) .

مثال/

```
$result = mysql_query ("select * from std") ;

$x = mysql_num_rows($result);

echo $x ;
```

- **الدالة mysqli_num_rows**

تستخدم هذه الدالة لحساب عدد الحقول التي تم جلبها من قاعدة البيانات .

مثال :-

```
$result = mysql_query ("select * from std") ;

$x = mysqli_num_rows($result);

echo $x;
```

- **الدالة mysql_affacted_rows**

تعود لنا هذه الدالة بعدد السجلات التي تأثرت من أحد العمليات (delete , insert , update) وتحتاج هذه الدالة إلى متغير واحد وهو مقبض الاتصال .

ملاحظة/ عند استخدام التعليمة delete بدون استخدام where فسوف يتم حذف جميع السجلات في الجدول ولكن الدالة سوف تعيد لنا القيمة صفر في هذه الحالة .

- **الدالة mysql_craete_db**

تستخدم هذه الدالة لإنشاء قاعدة بيانات جديدة ونحتاج فقط إلى اسم هذه القاعدة , وتعود بالقيمة true في حال نجاح العملية .

مثال :-

```
<?php
$link = mysql_connect ("localhost" , "root" , "  ")
or die ("فشل التصال") ;
if (mysql_create_db ("school") ) {
print ("تم إنشاء قاعدة البيانات") ; }
 else { print ("حدث خطأ") ; }
?>
```

- **الدالة mysql_drop_db**

تستخدم هذه الدالة لحذف قاعدة البيانات وجميع الجداول الموجودة فيها وتحتاج فقط إلى اسم قاعدة البيانات المطلوب حذفها , وتعود بالقيمة true في حال نجاح العملية .

مثال :-

```
<?php
$link = mysql_connect ("localhost" , "root" , "  ")
or die ("فشل التصال") ;
if (mysql_drop_db ("school") ) {
print ("تم إنشاء قاعدة البيانات") ; }
 else { print ("حدث خطأ") ; }
?>
```

- **الدالة mysql_list_dbs**

بواسطة هذه الدالة نستطيع معرفة قواعد البيانات المتوفرة لدينا ونحتاج فقط إلى مقبض الاتصال , لاحظ هذا المثال :-

```
$result = mysql_list_dbs ($link) ;
```

نلاحظ إن القيم المستخرجة وضعت في المتغير result ولكي نستفيد من هذه القيم نستخدم الدالة **mysql_fetch_row** التي تعرفنا عليهلمسبقاً .

- **الدالة mysql_list_tables**

بواسطة هذه الدالة نستطيع معرفة الجداول الموجودة في قاعدة البيانات وتحتاج هذه الدالة إلى متغير واحد هو اسم قاعدة البيانات .

مثال :-

```
$result = mysql_list_tables ("school") ;
```

نلاحظ بأن القيم المستخرجة وضعت في المتغير result ولكي نستفيد من هذه القيم نستخدم الدالة **mysql_fetch_row** التي تعرفنا عليهلمسبقاً .

- **الدالة mysql_list_fields**

بواسطة هذه الدالة نستطيع معرفة الحقول الموجودة في جدول معين وتحتاج هذه الدالة إلى متغيرين هما :-

1- اسم قاعدة البيانات .
2- اسم الجدول .

وسوف تعود هذه الدالة بمقبض للعملية لنستخدمه في معرفة معلومات حقول الجدول سواء الاسم أو النوع أو الطول أو الشرط , بواسطة الدوال التالية :-

1- الدالة mysql_field_name	لمعرفة اسم الحقل .
2- الدالة mysql_field_type	لمعرفة نوع الحقل .
3- الدالة mysql_field_len	لمعرفة طول الحقل .
4- الدالة mysql_field_flags	الشرط في الحقل .

- **الدالة mysqli_connect_error()**

تستخدم هذه الدالة لعرض الخطأ إذا لم يتم الاتصال بقاعدة البيانات بنجاح .

مثال/

```
die(mysqli_connect_error());
```

- **الدالة mysql_error( )**

تستخدم هذه الدالة لعرض الخطأ بالتفصيل عند التعامل مع قاعدة البيانات .

مثال/

```
die(mysql _error( ) ) ;
```

- **mysql_errno**

تستخدم هذه الدالة لتعيد آخر خطأ حصل في عملية التعامل مع قاعدة البيانات .

```
mysql _errno( مقبض الأتصال ) ;
```

الفصل العشرون

البرمجة كائنية التوجه (OOP)

ان الفائدة من البرمجة غرضة (كائنية) التوجة هو اختصار الوقت على المبرج فدلاً من ان يكتب كود برمجي من الصفر في كل مرة يريد فيها انشاء برنامج يمكنه كتابة مجموعة من الكلاسات والتي يمكن استخدامها في اكثر من برنامج (البرامج المتشابهه) ومن ثم عليه فقط استدعاء هذه الكلاسات في برنامجه .

ولانشاء كلاس يجب ان نتبع القواعد المخصصة لإنشائه والتعامل معه .

لإنشاء كلاس معين يجب أولاً ان نكتب كلمة class ثم بعدها اسم هذا الكلاس ونحن نختار الاسم كما نشاء لكن عند تسميته يجب ان نخضع لنفس قواعد تسمية المتغيرات بخلاف انه هنا لا يمكن ان نستخدم الفاصلة السفلية _ وبعد اسم الكلاس نضع قوس معكوف وبداخله نكتب كود الكلاس لاحظ :-

```
class  ahmed {
الكود هنا
}
```

في داخل الكلاس يمكن ان نكتب كود اعتيادي كما نكتب في خارج الكلاس حيث نستطيع ان نستخدم الدوال والمتغيرات والمصفوفات إلخ ... لكن هناك بعض التغييرات البسيطة التي نجريها على الكود وسنشرح هذه التغييرات بعد قليل , واذا اردنا اسدعاء الكلاس السابق نكتب الكود التالي :-

```
$x = new ahmed( );
```

لاحظ هنا اولاً قمنا بتعريف متغير اعتيادي زنعطيه أي اسم ثم نجعله يساوي new وبعدها اسم الكلاس المطلوب استدعائه , والآن وبعد ان عرفنا (استدعينا) الكلاس المطلوب يمكن ان نجلب أي قيمة من داخل هذا الكلاس وذلك بمجرد كتابة اسم المتغير الذي وضعنا فيه الكلاس ثم فاصلة وعلامة أكبر من هكذا -> بعدها اسم المتغير الذي نريد ان نجلب قيمته او اسم الدالة المطلوبة لكن لاحظ اننا هنا لا نكتب العلامة $ قبل اسم المتغير , لجلب قيمة متغير معرف في داخل الكلاس السابق نكتب هذا الكود :-

```
$x = new ahmed( );
$x -> var ;
$x -> fun( );
```

حيث ان var هو اسم متغير معرف في داخل الكلاس ahmed اما fun فهي دالة function معرفة في داخل نفس الكلاس , ويمكن ان نمرر بين قوسيها بارامترات اذا كنا قد وضعنا لها بارامترات في تعريفا ويمكن ان طبع قيمة المتغير اذا وضعنا قبله الدالة echo لاحظ :-

```
echo $x -> var;
```

القواعد التي يخضع لها الكود في داخل الكلاس

- **public**

نستخدم هذه الكلمة عندما نريد ان نعرف متغير او دالة بأنه عام أي انه يمكن استخدام هذا المتغير او الدالة خارج الكلاس ويمكن حتى ان نغير قيمته من خارج الكلاس .

مثال/

```
class  myClass {
public  $x = " Value " ;
public function myFun( ) {
}
}
```

ملاحظة/ يمكن تعريف المتغير بدون ان نعطيه قيمة ثم نسند له قيمة فيما بعد .

- **this**

عرفنا بأنه يمكن ان نتعامل مع المتغيرات والدوال من خارج الكلاس وذلك بذكر اسم الكلاس وبعده العلامة -> ثم اسم المتغير او الدالة لكن اذا اردنا ان نتعامل مع المتغير من داخل نفس الكلاس فسوف نحتاج الى كتابة الكلمة this

مثال/

```
class  myClass {
public  $x = " Value " ;
public function myFun( ) {
echo this -> x ;
}
public function myFun2( ) {
echo this -> myFun( ) ;
}
}
```

لاحظ بأنه اذا اردنا التعامل مع الدالة او المتغير من داخل نفس الكلاس فليس من الضرورة ان يكون من النوع public

ملاحظة/ يمكن ان نذكر اسم الكلاس بدلاً من كلمة this ولن تحدث أي مشكلة لكن يفضل ان نستخدم كلمة this عندما نريد الوصول الى متغير او دالة في داخل الكلاس وذلك لانه من المحتمل ان نريد تغيير اسم الكلاس الى اسم آخر وبذلك لن نضطر الى تغيير الاسماء القديمة للكلاس والتي ذكرناها في داخل نفس الكلاس .

- **private**

نستخدم هذه الكلمة عندما نريد ان نعرف متغير او دالة بأنه خاص أي لا يمكن استخدامه او الوصول اليه من خارج الكلاس ولا حتى من خلال توريث الكلاس (وسنشرح عملية التوريث بعد قليل) , واذا حاولنا الوصول اليه من خارج الكلاس سيعطينا خطأ .

مثال/

```
class  myClass {
private $x = " Value " ;
private function myFun( ) {
}
}
```

- **protected**

نستخدم هذه الكلمة عندما نريد ان نعرف متغير او دالة بأنه محمي أي لا يمكن استخدامه او الوصول اليه من خارج الكلاس ولكن يمكن الوصول اليه من خلال توريث الكلاس (وسنشرح عملية التوريث بعد قليل) , واذا حاولنا الوصول اليه من خارج الكلاس سيعطينا خطأ .

مثال/

```
class  myClass {
protected $x = " Value " ;
protected function myFun( ) {
}
}
```

ملاحظة / يمكن ان نستدعي المتغيرات الخاصة او المحمية من خارج الكلاس وذلك عن طريق ذكر هذه المتغرات في داخل دوال عامة .

مثال/

```
class  myClass {
protected $x = " Value " ;
public function myFun( ) {
return this -> x ;
}
}
$obj = new myClass( );
echo $obj -> myFun( ) ;
```

هنا سيتم طباعة القيمة value

- **const**

يتم تعريف الثابت بإستخدام الكلمة المحجوزة const ويفضل أن يكون اسم الثابت بالحروف الكبيرة والثابت يكتب بدون العلمة $ ويجب إعطاء الثابت قيمة عند تعريفة ولا نستطيع تغيير هذه القيمة فيما بعد لانها ثابته .
عندما نريد إستخدام الثابت ل نستخدم طريقة إنشاء كائن من الفئة كما سبق ولكن نقوم بكتابة اسم الكلاس ثم العلامتين :: ثم اسم الثابت .
مثال/

```
class myClass {
const NAME = "User";
}
echo myClass :: NAME ;
```

واذا اردنا ان نستخدم الثابت في داخل نفس الكلاس فيمكن ان نكتب اسم الكلاس او الكلمة المحجوزة self (لا يمكن استخدام الكلمة this مع الثابت) وبعدها نضع العلامتين :: ثم اسم الثابت .
مثال /

```
class myClass5 {
const NAME="User";
public function test( ){
return self::NAME;
}

echo myClass5::NAME;
echo "<br>";
$obj=new myClass5( );
echo $obj->test( );
```

- **static**

نستخدم هذه الكلمة عندما نريد ان نعرف متغير بأنه ساكن , والمتغير من لنوع الساكن يظل محتفظ بقيمتة داخل الفئة (الكلاس) إلى أن ينتهى عمل الفئة (الكلاس) وهو يشبة فى هذا عمل المتغيرات التى تعرف فى بداية الفئة ولكن فى بعض الحيان نحتاج لتعريف متغيرات داخل الدوال ونريد أن تظل قيمتها محفوظة داخل المتغير ولا تنتهي بإنتهاء عمل الدالة ولهذا نقوم بتعريف المتغير على أنه ساكن , ولهذا تستخدم static لتعريف المتغير داخل الدوال .
وأيضاً لاحظ تجاهل القيمة الابتدائية التى تسند للمتغير الساكن عند تعريفة داخل الدالة

```
<?php
class myClass6 {
public function test(){
$t=0;
$t++;
return $t;
}
public function test2(){
static $t2=0;
$t2++;
return $t2;

}
public function test3(){
echo $this->test()."<br>";
echo $this->test()."<br>";
echo $this->test()."<br>";
echo "----------<br>";
echo $this->test2()."<br>";
echo $this->test2()."<br>";
echo $this->test2()."<br>";
}
}
$obj=new myClass6();
$obj->test3();
?>
```

ستكون النتيجة بهذا الشكل

```
1
1
1
----------
1
2
3
```

تعريف المتغيرات الساكنة داخل الدوال يجعلنا لا نستطيع إستخدام هذه المتغيرات خارج النطاق المعرفة به لانها تعتبر متغيرات محلية ذات طابع خاص , ولتعريف متغيرات عامة من النوع الساكن يتم الوصول إليها من خلال الكلمة المحجوزة self ثم العلامتين :: ثم اسم المتغير الساكن ولا ننسى علامة $ خلاف المتغير الثابت .

ولاستخدام المتغيرات الساكنة خارج نطاق الفئة نكتب اسم الفئةمتبوعاً بالعلامتين :: ثم اسم المتغير الساكن .

مثال/

```
class myClass7 {
public static $name="user1";
private static $name2="user2";
public function test(){
return self::$name2;
}
}
echo myClass7::$name;
echo "<br>";
$obj=new myClass7();
echo $obj->test();
```

الدوال الساكنة

يتم تعريف الدوال على أنها ساكنة ولكن فى هذه الحالة لا نستطيع إستخدام الكلمة this أي لا نستطيع إستخدام مكونات الفئة (الكلاس) داخلها ولكن يمكن إستخدام المكونات المعرفة على أنها ساكنة داخل الدوال الساكنة , ويمكن إستخدام المكونات الساكنة داخل الفئة (الكلاس) بكتابة self ثم :: ثم اسم العنصر الساكن , وعند إستدعاء الدالة خارج الفئة نكتب اسم الفئة ثم :: ثم اسم الدالة .

مثال /

```
class myClass8 {
public static $name="user";
public static function test( ){
return self::$name;
}
public static function test2( ){
return self::test( );
}
}
echo myClass8::test2( );
```

الوراثة extends

الاسم يشرح نفسه فهى عملية وراثة مكونات الفئة (الكلاس) الموروثة فى الفئة الوارثة أي إمكانية إستخدام مكونات الفئة الموروثة – من دوال ومتغيرات – في الفئة (الكلاس) الوارثة .

ملاحظة/ يتم وراثة المكونات المعرفة على أنها عامة أو محمية ولا يتم توريث المكونات المعرفة على أنها خاصة .

مثال/

```
<?php
class A {
public $name='user1';
private $name2='user2';
protected $name3='user3';
}
class B extends A {
public function test(){
echo $this->name;
echo "<br>";
//echo $this->name2;
echo $this->name3;
}
}
$obj=new B();
echo $obj->test();
echo "<br>";
echo $obj->name;
?>
```

الكود الموضوع فى التعليق هو كود خاطئ لانه يعتبر عملية وصول لمتغير خاص والمكونات المعرفة على أنها خاصة لا تورث , وعند إنشاء كائن من الفئة الجديدة نستطيع إسخدام مكونات الفئة ومكونات الفئة التى ورثتهأيضاً بشرط أن تكون معرفة على أنها مكونات عامة .

ملاحظة/ لا يمكن وراثة الكلاس الذي ورث من كلاس آخر مثلاً في المثال السابق لا يمكن وراثة الكلاس B

ملاحظة/ نستخدم الكلمة المحجوزة this للوصول لمكونات الفئة (الكلاس) الموروثة ولكن فى حالة الثوابت والمكونات المعرفة على أنها ساكنة يتم إستخدام الكلمة المحجوزة parent ثم يتبعها :: ثم اسم الثابت أو العنصر الساكن ملحوظة :" الثابت لا تسبقة العلامة $ ولكن تسبق العناصر الساكنة " .

مثال /

```
<?php
class A {
const NAME="User1";
public static $name='user1';
private static $name2='user2';
}
class B extends A {
const NAME2="User2";
private static $name3='user3';
```

```
public function test( ){
echo parent::NAME;
echo "<br>";
echo parent::$name;
echo "<br>";
echo self::NAME2;
echo "<br>";
echo self::$name3;
}
}
echo B::NAME;
echo "<br>";
echo B::$name;
echo "<br>";
echo B::NAME;
echo "<br>";
$obj = new B( );
echo $obj -> test( );
?>
```

- **abstract**

توضع هذه الكلمة قبل اسم اللاس اذا كنا نريد ان نكتب كلاس وهذا الكلاس لا نريد ان نستخدمه إلا في التوريث .

مثال/

```
abstract class A {
public function test( ){
echo "user";
}
}
class B extends A {
public function test2( ){
$this->test( );
}
}
//$obj=new A( );
//echo $obj->test( );
$obj=new B( );
echo $obj->test2( );
```

لاحظ ان الكود الموجود في التعليق هو خاطئ .

- **final**

نستخدم هذه الكلمة قبل اسم الكلس اذا كنا لا نريد ان نسمح لهذا الكلاس ان يورث من قبل كلاس آخر .
مثال/

```
final class my{
public function test( ){
echo "user";
}
}
```

ملاحظة/ يمكن استخدام final قبل اسم دالة function لنعلن ان هذه الدالة نهائية .

- **interfaces**

نستخدم هذه الكلمة بدلاً من اسم كلمة كلاس اثناء تعريف الفئة ولا يمكن ان نصل الى محتوياته في ما بعد إلا من داخل كلاس آخر حيث هنا في داخل الـ interfaces نضع التعليمات او الخطة التي نريدها ان تطبق في ما بعد على الكلاس الذي سنستدعيها فيه
مثال/

```
interfaces MyName {
public function  test( );

}
```

كما تلاحظ وضعنا بعد الـ interfaces اسم نحن نختاره وفي داخله نضع الكود المطلوب , ولاحظ اننا عرفنا الدالة ولكن لم نضع لها قوسين معروفين ولم نضع بداخلها كود وسوف نضع لها الكود من داخل الكلاس الذي سوف يستدعي هذا الـ interfaces , ويتم استدعاءه من خلال الكلمة implements .
مثال/ لأستدعاء القاعدة في المثال السابق

```
class ahmed implements MyName  {
public function  test( ) {
echo "The Value"; }
}
$n = new ahmed( );
$n -> test( );
```

لاحظ اننا استخدمنا نفس اسم الدالة السابقة test وهذا الامر اجباربي أي اننا مجبرون على استخدام نفس الاشياء المعرفة في داخل القاعدة interfaces في داخل الكلاس الذي استدعاها , ويمكننا في نفس الوقت ان نعرف دوال آخرى جديدة ,لكن اذا لم نستخدم الدوال الموجودة في القاعدة interfaces سوف يتوقف الكلاس عن العمل ويعطي خطأ .

ملاحظة/ يمكن للـ interfaces ان يرث interfaces آخر .

مثال /

```
interface A {
public function test( );
}
interface B {
public function test2( );
}
interface C extends A {
public function test3( );
}
```

ملاحظة / يمكن للكلاس ان يستدعي أكثر من interfaces واحد ونفصل بينهم بفاصلة

مثال/

```
<?php
interface A {
public function test( );
}
interface B {
public function test2( );
}
interface C extends A {
public function test3( );
}
class D implements B,C{
public function test( ){
echo "test";
}
public function test2(){
echo "test2";
}
public function test3(){
echo "test3";
}
}
$obj=new D( );
$obj->test( );
echo "<br>";
$obj->test2( );
echo "<br>";
$obj->test3( );
?>
```

- **السمات trait**

السمات هى عبارة عن طريقة للتخلص من القيود التى فرضاتها الوراثة الفردية وأعنى بالوراثة الفردية هى أن لغة php ل تدعم الوراثة المتعدده كما فى لغة c++ لن الوراثة المتعددة على رغم قوتها فى تسبب كثير من المشاكل والتعقيد ولهذا أنتجت php ما يعرف بالسمات .
يتم تعريف السمة من خلل الكلمة المحجوزة trait ويتم استخدام السمات في الفئة من خلال الكلمة المحجوزة use .
مثال/

```
<?php
trait A{
public function test() {
return "user1";
} }
trait B{
public function test2() {
return "user2";
} }
class C{
use A,B;
}
$obj = new C();
echo $obj->test();
echo "<br>";
echo $obj->test2();
?>
```

في هذا المثال كأنما قام الكلاس C بوراثة السمتين A و B , وهذه مشابهه لعملية الوراثة المتعددة .

- **__construct و __destruct**

تستخدم الدالة __construct للتشغيل التلقائي وهي عبارة عن دالة محجوزة .
مثال/

```
class A{
public function __construct( ) {
echo "Welcome";
} }
$n = new A( );
```

لاحظ بأنه سيطبع الكلمة Welcome بدون ان نطلب منه تنفيذ الدالة أي بمجرد تعريف الكلاس تم تنفيذ الدالة التي قد اعطيناها امر التنفيذ التلقائي .
وتستخدم الدالة __destruct لأنهاء التشغيل التلقائي .
ملحوظة : دالة البناء (التشغيل التلقائي) يمكن أن تأخذ قيم (وسيط) ويتم تمرير هذه القيم إليها عند إنشاء كائن من الفئة (الكلاس) , ودالة الهدم لا تأخذ أي قيم كوسيط .
مثال/

```
class A{
public $name;
private $name2;
protected $name3;
public function __construct($n1,$n2,$n3,$n4){
$this->name = $n1;
$this->name2 = $n2;
$this->name3 = $n3;
echo $this->name;
echo "<br>";
echo $this->name2;
echo "<br>";
echo $this->name3;
echo "<br>";
$this->test($n4);
}
public function __destruct(){
echo "<br>";
echo "yossef";
}
public function test($n){
echo $n;
}
}
$obj = new A("user1","user2","user3","user4");
```

ملاحظة/ يمكن ان نعرف متغير ونضع داخله تعريف كلاس معين كما كنا نفعل في السابق لكن من داخل نفس الكلاس .
مثال/

```
class  MY{
public $x = new MY( );
}
```

ويمكن ان نضع كلمة self بدلاً من اسم الكلاس .

- **__call**

في بعض الأحيان عندما نستدعي كلاس معين ونريد ان نجلب قيم من بعض الدوال الموجودة فيه قد نخطأ مثلا في اسم الدالة او في البارامترات الممررة الى الدالة فبالنتيجة سيطبع لنا المتصفح خطأ عند التنفيذ لكنه لن يخبرنا ما هو الاسم الذي أخطأنا فيه وأين الخطأ بالضبط وهنا تظهر فائدة هذه الدالة السحرية __call وكل ما علينا هو تعريف هذه الدالة في داخل الكلاس , ولاحظ ان هذه الدالة تأخذ بارامترين الأول هو $fname وهو مخصص لاسم الدوال والثاني هو $farray مخصص للبارامترات ولاحظ بأنه ستعود اسماء البارامترات على شكل مصفوفة .

ملاحظة/ في البرمجة الكائنية تعتبر كل الدوال التي تبدأ بالعلامتين (__) دوال سحرية مع العلم بأنه يمكننا أن ننشأ دوال تبدأ بهذه العلامتين .

ملاحظة/ يمكن ان نظهر اسم الكلاس الذي حدث فيه الخطأ من خلال __CLASS__

مثال/

```
class MyClass{
public function __call($fname , $farray){
echo  $fname . " -- " .  __CLASS__ ;
echo print_r($farray);
}
public function test($var){
echo $var
}
}
$ob = new MyClass( );
$ob->tets("Ahmed");
```

لاحظ بأننا هنا أخطأنا في كتابة اسم الدالة لكن سيظهر لنا المتصفح اسم الدالة التي أخطأنا فيها وهي tets عند تنفيذ الكود, وبعدها سيظهر اسم الكلاس الخطأ والذي هو في مثالنا MyClass , ولو كنا قد أخطأنا في البارامترات فأنه سيظهر لنا اسماء البارامترات التي اخطأنا فيها .

ملاحظات عامة

أولاً :- التعليقات :- يحتاج المبرمج إلى التعليقات لكي يوضح البرنامج ويكون مفهوم عند قراءة الكود , لكن هذه التعليقات لن تظهر ولن تؤثر على البرنامج , ويمكن كتابة هذه التعليقات بطريقتين

أ- إذا كان التعليق في سطر واحد يمكن وضع (//) او العلامة (#) قبل التعليق .

ب- أما إذا كان التعليق يتكون من أكثر من سطر (عدة أسطر) فنضع قبل بداية التعليق (/*) وفي نهاية التعليق (*/) .

```
<?php
echo 88;
// تعليق بسطر واحد
echo " Hi 77 ";
/*
تعليق بعدة سطور
*/
?>
```

ثانياً :- يمكن كتابة الدوال بالحروف الكبيرة أو الصغيرة فليس هناك مشكلة حيث لا فرق بين **echo** و **ECHO** , لكن انتبه فبالنسبة إلى المتغيرات فالأحرف الكبيرة ليست كالصغيرة .

ثالثاً :- بالنسبة إلى الدالة echo فيمكن أن نضيف جزء إلى جزء آخر أي نص إلى متغير أو إلى وسم في الدالة عن طريق إضافة نقطة (.) وهي تشبه عملية الجمع , لاحظ هذا المثال :-

```
echo "Hi"."welcome"." ".Ahmed";
```

لتكون النتيجة هكذا Hi welcome Ahmed , لاحظ أن طباعة العلامتين (" ") فارغتين يؤدي إلى طباعة فراغ (فاصلة) .

وأيضا يمكن إضافة المتغيرات النصية إلى بعضها , مثال :-

```
$x = "Ali";
$y = "Ahmed";
$z = $x . $y ;
```

```
echo $x . $y ;
```

لتكون النتيجة هكذا Ali Ahmed

أو يمكن إضافتها بهذا الشكل :-

```
$b = "My" ;

$b. = "program"

echo $b ;
```

لتكون النتيجة My program

رابعاً :- يمكن استخدام وسوم لغة html مع لغة php بوضعها داخل الدالة **echo** وبين علامتي تنصيص (" ")

مثال :-

```
echo "<h1> welcome </h1> <br> Hi ";

echo "<p>";
```

حيث إن الوسم **<p>** يعمل على ترك سطرين فارغين .

خامساً :- يمكن كتابة نص (قيمه نصية) ونضعها بين علامتي تنصيص مفردة (' ') أو مزدوجة ولكن إذا كانت علامة التنصيص مفردة في البداية فلا يجوز أن تنتهي بعلامة تنصيص مزدوجة ولا يجوز العكسأيضاً .

لا يجوز وضع علامة التنصيص المستخدمة في النص من نفس النوع , لاحظ هذه الأمثلة :-

```
$x= " مثال " خاطئ " ;
$x= ' مثال ' خاطئ ' ;
$x= " مثال ' صحيح " ;
$x= ' مثال " صحيح ' ;
```

وأيضاً يمكن إظهار علامة التنصيص في النص من خلال إضافة العلامة (\ ) قبل علامة التنصيص , وإن أردنا إظهار شكل هذا الرمز يمكن كتابيه مرتين (\\ ) , مثال :-

```
$x = " Hi \" welcome " ;
```

سادساً :- هناك بعض الأحرف لا يستطيع المتصفح إضافتها إلى عنوان الصفحة بصيغتها الحقيقية بل يستخدم لغة التشفير (URL Encoding) في التعريف عنها وهذا جدول بالرموز التي يستخدمها المتصفحبدلاً من عرضها بصورتها الحقيقية :-

الحرف (المفتاح)	شفرته	الحرف (المفتاح)	شفرته
Tab	%9	,	%2C
Space	%20	.	%2E
!	%21	/	%2F
"	%22	:	%3A
#	%23	;	%3B
@	%40	<	%3C
\	%5C	>	%3E
(	%28	=	%5D
)	%29	؟	%3F
+	%2B	%	%25
&	%26		

سابعاً :- للربط بين لغة html ولغة php (أو إرسال البيانات من صفحة html التي فيها النماذج وصفحة php التي تستقبل البيانات) سنوضح ذلك في هذا المثال التطبيقي :-

1- سوف نعمل صفحة html ونكتب فيها هذا الكود

اسم صفحة ال (php)

```
<html dir = "rtl" >
<form  method = "get" action = "text.php" >
ما هي وجبتك المفضلة في الصباح ؟                ( اسم المتغير )
<br>
<input  type = "text" name = "food" value = " جبنه ومربى " >
<br>
```

```
<input  type = "submit" value = " إرسال " >
<input  type = "reset" value = " مسح " >
</form>
</html>
```

سوف نحفظ هذه الصفحة في السيرفر المحلي باسم (textbox.html)

2- ثم ننشأ صفحة php ونكتب الكود التالي

```
<?php                                   المتغير في html
echo " وجبتك المفضلة هي " . " " . $food ;
?>
```

الآن سوف نحفظ الصفحة في السيرفر المحلي باسم (textbox.php) وبعد ذلك نذهب إلى المستعرض ونكتب عنوان صفحة html هكذا :- http//localhost/textbox.html بعد ذلك نكتب وجبتنا المفضلة وإرسالها , سوف تلاحظ ظهور النتيجة في العنوان وذلك لأننا استخدمنا الخاصية (**get**) ولكن ستكون النتيجة في العنوان مشفرة لأننا استخدمنا الحروف العربية .

ملاحظة/ يمكن ان نرسل البيانات من نموذج الأدخال الى صفحة الـ php على شكل مصفوفة وذلك فقط بكتابة قوسي المصفوفة [ ] بعد الاسم

```
<input type="text" name="my_name[ ]">
<input type="text" name="my_name[ ]">
```

ولاحظ اننا اعطينا لكلاهما نفس الاسم لأنهم سيكونون في نفس المصفوفة جيث سيكون الأول يحمل المرتبة 0 والثاني يحمل المرتبة 1 وهكذا اذا كان لدينا عناصر اكثر , سوف نستقبل البيانات في صفحة الـ php بهذا الشكل

```
$_POST[my_name][0];
$_POST[my_name][1];
```

او يمكن استقبال البيانات بشكل آخر هكذا

```
$arr = $_POST[my_name] ;
echo arr[0];
echo arr[1];
```

ثامناً :- يمكن تحديد عملية المساواة بين متغيرين بوضع علامة (==) حيث إنها تكون صحيحة إذا كانت القيم متساوية ولا تهتم أن تكون المتغيرات من نفس النوع أو لا , لاحظ هذه المثال :-

```
$x = " 10 " ;

$y = 10 ;

if ( $x == $y ) { echo " القيم متساوية " ; }
```

لاحظ النتيجة سوف تكون متساوية وتحقق الشرط على الرغم من أن المتغير x يحمل قيمة نصية والمتغير y يحمل قيمة عددية , وإن أردنا أن تكون المساواة بين قيم المتغيرات وكذلك بين نوع المتغيرات سوف نحتاج أن نكتب المساواة المضاعفة (===)بدلاً من المساواة المزدوجة (==)

تاسعاً :- لاحظ أن (**break**) تعمل على الخروج من العبارة الشرطية الموجودة فيها , أما (**exit**) فتعمل على الخروج من كامل الكود لاحظ هذين المثالين التوضيحيين :-

مثال 1 :-

```
$x = 10 ;
if ( $x = 10 ) { echo " 10 " ;
exit ; }
else if ( $x < 11 ) { echo " small " ; }
else { echo " big " ; }
```

مثال 2 :-

```
$x = 10 ;
if ( $x = 10 ) { echo " 10 " ;
break  ; }
else if ( $x < 11 ) { echo " small " ; }
else { echo " big " ; }
```

عاشراً :- المسارات في php

لنفترض أننا نود التعامل مع الملف "file.txt" فيكون لدينا أحد الحالات التالية

1- أن يكون في نفس المجلد الذي ننفذ فيه برنامجنا وهنا نستطيع كتابة اسم الملف فقط دون الحاجة لأي إضافات , مثال /

```
"file.txt"
```

2- أن يكون الملف في مجلد نفرض أن اسمه data داخل المجلد الموجود فيه برنامجنا , ففي هذه الحالة نكتب مسار الملف كما يلي

```
" ./data/file.txt"
```

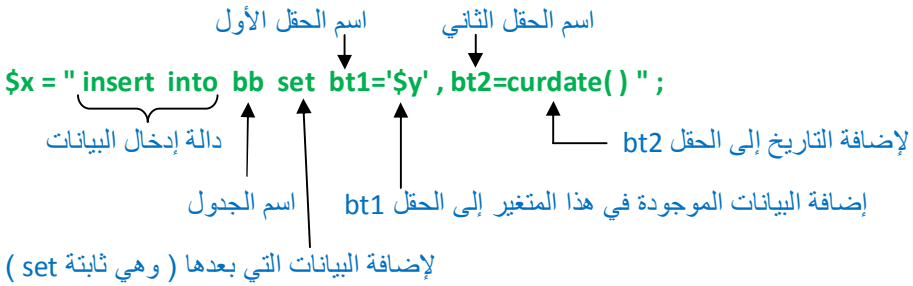
3- أن يكون الملف في المجلد الأب الموجود فيه برنامجنا , وهنا يكون المسار كالتالي

```
" ../file.txt"
```

4- أن نحدد المسار كعنوان انترنت

```
"www.example.com/file.txt"
```

أحدعشر :- لإضافة بيانات إلى حقول في الجدول



إثنى عشر :- إذا أنشأنا نموذج إدخال في الـ html فيجب أن نحدد الصفحة التي نريد أن نعالج البيانات التي يدخلها المستخدم وذلك من خلال الخاصية action والتي هي إحدا خصائص الوسم form لكن إذا أردنا أن نعالج البيانات في نفس الصفحة التي فيها نموذج الأدخال فيمكن أن لا نضع الخاصية action و لكن يفضل وضع هذه الخاصية واضافة نفس اسم الصفحة التي فيها النموذج (وهي نفسها الصفحة التي سنعالج البيانات فيها) لكن لو فرضاً تم تغيير اسم الصفحة فيما بعد فسيحدث خطأ بكل تأكيد ولتلافي هذه المشكلة سنجعل قيمة الخاصية action تساوي $_SERVER[‘ PHP_SELF ’] وهذه التعليمة تعني أن يتم المعالجة في نفس الصفحة .

مثال/

```
<form  action=” <?php $_SERVER[‘ PHP_SELF ’] ?> ”>
```